



Approximate Newton Methods for Distributed Learning over Communication-Constrained Wireless Networks

Ganesh Sharma

*A thesis submitted in fulfillment of the requirements
for the Ph.D. degree in Data Science*

at the

Hamilton Institute
Maynooth University
Maynooth, Co. Kildare, Ireland

under the supervision of

Prof. Subhrakanti Dey
Dept. of Electrical Engineering
Uppsala University

August 2025

Summary

Traditional Machine Learning (ML) methodologies typically involve aggregating datasets on a central server for analysis and model training. While effective in certain contexts, this centralized approach presents significant limitations, particularly concerning data sensitivity, privacy, and security. These constraints hinder the full realization of ML's potential, thereby slowing progress across a range of applications.

Distributed Machine Learning (DML) offers a promising alternative by decentralizing the learning process. In this paradigm, data remains at its source, while learning models are transmitted to the data, thereby eliminating the need for data extraction. Federated Learning (FL), a prominent subset of DML, is specifically designed to address challenges related to data privacy. FL commonly employs distributed stochastic gradient descent (DSGD) for optimization, yet it faces several practical challenges, including slow convergence and high communication overhead.

This thesis investigates enhanced alternatives to conventional FL through the application of Hessian-based optimization methods. By leveraging second-order information, the learning process can be accelerated, requiring fewer iterations and reduced communication to accomplish learning tasks efficiently.

The work addresses key challenges in FL and Fully Distributed Learning (FDL) over wireless networks, with a particular emphasis on minimizing communication costs while exploiting the advantages of second-order optimization.

In the FL setting, we propose Distributed Approximate Newton with Determinantal Averaging (DANDA), a Newton-type method that significantly reduces the number of communication rounds required for convergence. To accommodate the limited computational capabilities of client devices, we incorporate approximation techniques for Hessian computation. DANDA operates over over-the-air Multiple Access Channels (MAC), and its performance is analyzed under realistic wireless conditions with channel fading and noise.

Building on this, we develop *Lazy-DANDA* and *Lazy-DANTA*, approximate Newton-based algorithms tailored for fading MAC environments. These methods integrate subsampled Hessians, weighted Hessian averaging, and an adaptive Hessian update strategy that transmits updates only when necessary, thereby further reducing communication

overhead. To counteract distortions from channel fading, we incorporate channel inversion and power control mechanisms, preserving signal quality while regulating power consumption.

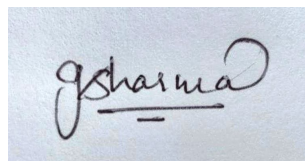
In the FDL scenario, we extend the server-dependent GIANT algorithm into Network-GIANT, enabling decentralized node-to-node learning without a central server. This is achieved by combining gradient tracking, Newton-type iterations, and consensus-based averaging of Newton updates. Network-GIANT achieves semi-global exponential convergence under strong convexity and smoothness assumptions, addressing the slow convergence issue inherent to fully distributed setups. We further refine this approach into Network Exact Convergence-GIANT, which employs finite-time distributed consensus to match the exact convergence properties of GIANT in the FL setting.

Collectively, these contributions advance the state of the art in second-order optimization for FL and FDL, delivering faster convergence, reduced communication overhead, and improved robustness under realistic wireless network conditions.

Declaration

I, Ganesh Sharma, declare that this thesis titled, “**Approximate Newton Methods for Distributed Learning over Communication-Constrained Wireless Networks** ” and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. Except for such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- I used external tools (Grammarly, ChatGPT 4) for assistance strictly limited to grammar, syntax, and spelling checks, as well as for identifying potential ambiguities or difficult-to-read sentences. At no point did these tools contribute to the original research, ideas, analytical conclusions, or text body presented within this thesis.



Signed: _____

Date: Aug-2025

Acknowledgements

“Learning gives creativity, creativity leads to thinking, thinking provides knowledge,
knowledge makes you great.”

Dr. APJ Abdul Kalam

Completing this PhD program has been a transformative journey, and I owe immense gratitude to the numerous colleagues, friends, and family members whose unwavering support and guidance made it all possible.

First and foremost, I extend my heartfelt appreciation to my PhD supervisor, Prof. Subhrakanti Dey, whose consistent mentorship and encouragement have been instrumental throughout these past four years. The lessons learned under his guidance are invaluable, as are the skills honed during this period. Professor Subhra’s genuine concern for his students and his prioritization of their interests have left an indelible mark on me. I consider myself truly fortunate to have had the opportunity to learn from him.

I also wish to extend special thanks to Professor Luca Schenato, my mentor and primary collaborator, whose indispensable guidance and patience were invaluable during my PhD journey. I am particularly grateful to him for facilitating the unforgettable research internship at the Department of Information Engineering, University of Padova, Italy, in 2022. The memories from my time spent in Padova, especially the cherished moments in the Dolomites and Florence with my PhD colleagues Bharvi Dhall and Amit Chinwan, will always hold a special place in my heart.

Acknowledgment is also due to the funding source of my PhD, the “SFI Center for Research and Training in Foundations of Data Science,” which provided invaluable opportunities to collaborate with esteemed researchers worldwide. I extend my gratitude to all three Directors of CRT, namely Prof. David Malone (and Prof. Ken Duffy), Prof. Claire Gormley, and Prof. James Gleeson. Additionally, I express sincere appreciation to Janet Clifford (CRT), Rosemary Hunt, Kate Marie Moriarty, and Joanna O’Grady (Hamilton Institute) for their unwavering administrative support throughout my PhD journey.

Lastly, I express profound gratitude to my family for their unconditional love, support, and encouragement throughout every adventure and challenge I undertook. Each member

of my family, including my father, mother, and beloved sisters, played an indispensable role in my journey.

I cannot conclude without expressing gratitude to Bharvi Dhall and Amit Chinwan, whose unwavering support and companionship were a constant source of strength throughout these past years, through both the highs and lows.

This thesis has emanated from research conducted with the financial support of Science Foundation Ireland under Grant number 18/CRT/6049.

List of Publications

Peer-reviewed Conference papers

- Ganesh Sharma and Subhrakanti Dey. “On analog distributed approximate Newton with determinantal averaging,” In 2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC), pages 1–7. IEEE, 2022.
- Maritan, Alessio, Ganesh Sharma, Luca Schenato, and Subhrakanti Dey. “Network-GIANT: Fully distributed Newton-type optimization via harmonic Hessian consensus.” In 2023 IEEE Globecom Workshops (GC Wkshps), pp. 902-907. IEEE, 2023.
- Maritan, Alessio, Ganesh Sharma, Subhrakanti Dey, and Luca Schenato. “Fully-distributed optimization with Network Exact Consensus-GIANT.” In 2024 IEEE 25th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 436-440. IEEE, 2024. (*shortlisted for the best student paper award*).

Peer-reviewed Journal articles

- Ganesh Sharma and Subhrakanti Dey. “Over-the-air Lazy-Hessian-based Federated Learning”. (In preparation for submission).

Contents

1	Introduction	1
1.1	Background	1
1.2	Federated Learning	2
1.2.1	Challenges	4
1.2.2	Literature Review	6
1.3	Fully Distributed Learning	9
1.3.1	Challenges	10
1.3.2	Literature Review	11
1.4	Research Objectives	12
1.4.1	Resource-Efficient Communication	12
1.4.2	Fast Convergence with lower Computational complexity	12
1.4.3	Node selection and Power control	13
1.4.4	Improved Convergence in Fully Distributed Learning	13
1.5	Thesis Contributions	13
1.6	Thesis outline	14
2	Distributed Approximate Hessian with Determinantal Averaging	16
2.1	Introduction	16
2.1.1	Related Work	16
2.1.2	Our Contributions	18
2.2	System Model	19
2.2.1	Distributed Learning Model	19
2.2.2	Communication Model	19
2.3	Distributed Newton algorithms with Determinantal Averaging	20
2.3.1	Distributed Newton with Determinantal Averaging: DNDA	20
2.3.2	DNDA with Recursive Least Squares	23
2.3.3	Distributed Approximate Newton with Determinantal Averaging: DANDA	24
2.4	Experimental Results	25

2.5	Conclusions	31
3	Over-the-air Lazy-Hessian based Federated Learning	32
3.1	Introduction	32
3.1.1	Related Work	32
3.1.2	Our Contributions	33
3.2	System Model	34
3.3	The Algorithm	35
3.3.1	Newton Direction Estimation	36
3.3.2	Node Scheduling and Power Control	36
3.3.3	Lazy-Approximate Hessian	40
3.3.4	The Algorithm: Lazy-DANCIDA/Lazy-DANCITA	41
3.4	Convergence Analysis	42
3.5	Experimental Results	43
3.6	Conclusions	46
4	Fully distributed Newton-type optimization	47
4.1	Introduction	47
4.1.1	Related Work	47
4.1.2	Our Contributions	49
4.2	System Model	49
4.3	Fully Distributed GIANT	51
4.3.1	Network-GIANT	52
4.3.2	NEC-GIANT	53
4.4	Convergence Analysis	55
4.4.1	Network-GIANT	56
4.4.2	NEC-GIANT	59
4.5	Experimental Results	60
4.6	Conclusion	63
5	Conclusions and Future Work	65
6	Appendix A	68
6.1	Proof of Theorem 3.4.2	68
	Bibliography	73

List of Figures

1.1	FL	3
1.2	Fully Distributed Learning	10
2.1	Transmission Scheme over a wireless MAC	18
2.2	Simulation results on (a) convergence and (b) computation time averaged over 100 channel realizations performed on a 1.6 GHz Dual-Core Intel Core i5 processor machine with 8 GB 2133 MHz LPDDR3 RAM	26
2.3	Simulation results for (a) Convergence performance of DANDA with different power adjustment factor (PAF) values for DANDA and (b) Power requirements vs convergence rate performance of DANDA averaged over 100 channel realizations	26
2.4	Convergence comparison of DANDA with different rank values, keeping every other parameter constant	28
2.5	Test data performance of DANDA, DNDA, GBMA, and LN	29
2.6	Total power vs average iterations comparison of DANDA, DNDA, LN, and GBMA	30
2.7	Convergence performance of DNDA, DNDA with RLS, and a combination of both	31
3.1	Transmission Scheme	35
3.2	Simulation results on convergence and computation time averaged over 100 channel realizations	44
3.3	Simulation results showing test accuracy of distributed algorithms	44
3.4	Simulation results showing Power required up to convergence for the distributed algorithms	46

4.1	The first three figures from the left show the convergence performance on the training set, while the last figure displays the transient evolution of the accuracy on the test set of the MNIST dataset. The convergence is plotted against multiple metrics, namely the number of iterations, the overall computation time, and the overall communication cost in bits of the network.	60
4.2	The first three figures from the left show the convergence performance on the training set, while the last figure displays the transient evolution of the accuracy on the test set of the Coverttype dataset. The convergence is plotted against multiple metrics, namely the number of iterations, the overall computation time, and the overall communication cost in bits of the network.	61
4.3	The figure shows a convergence comparison of centralised Newton, GIANT, and Network-GIANT, where k_1 indicates consensus rounds to determine global gradient, w_i^{k+1} , and k_2 indicates consensus rounds to determine global parameter vector, θ_i^{k+1} , and NEC-GIANT. In both the figures, the first plot shows training loss vs iteration, and the second plot shows training loss vs machine time in seconds.	63

List of Tables

1	List of Symbols and Their Descriptions	xii
2.1	List of Simulation Parameters	27
2.2	Average computer time (seconds) taken by algorithms per iteration per node on a 1.6 GHz Dual-Core Intel Core i5 processor machine with 8 GB 2133 MHz LPDDR3 RAM	28
4.1	Communication cost of different second-order fully distributed algorithms, quantified as the number of scalars transmitted along each active edge of the network (i.e., for each non-zero off-diagonal entry of the mixing matrix P) at each iteration. The cost is expressed as a function of the dimension d of the problem and the number K of consensus steps.	54

List of Notation

Symbol	Description
Indices and Sets	
n	Node index
k	Iteration index
\mathcal{R}	Entire dataset
\mathcal{S}_n	Dataset at node n
S_n	Sub-sampled dataset at node n
Scalars and Parameters	
N	Number of nodes
d	Number of features
β	Learning rate
a_n	Determinant of local Hessian at node n
$h_{n,k}$	Channel gain for node n at iteration k
R	Size of entire dataset
b_n	Size of dataset at node n ($ \mathcal{S}_n $)
P	Power Adjustment Factor (PAF)
P_k	PAF at k^{th} iteration
$P_{n,k}$	local PAF at n^{th} node and k^{th} iteration
α	Power Distribution Factor (PDF), $0 < \alpha < 1$
P_0	Power Constant
ϕ_h	Channel gain threshold
Vectors and Matrices	
\mathbf{g}_n	Local gradient at node n
\mathbf{G}	Global gradient
\mathbf{H}_n	Local Hessian at node n
\mathbf{H}	Global Hessian
$\boldsymbol{\theta}$	Parameter vector
$\boldsymbol{\theta}^*$	Optimal parameter vector
\mathbf{w}_k	Thermal noise at iteration k , $\mathbf{w}_k \sim \mathcal{N}(0, \sigma_w^2 \mathbf{I}_d)$
$\mathbf{s}(t)$	d orthogonal normalized waveforms

Table 1: List of Symbols and Their Descriptions

1

Introduction

This study focuses on communication-efficient second-order optimization methods for FL and FDL in wireless network environments. In traditional ML setups, models are typically trained on data aggregated in a single location. The rapid proliferation of connected devices has led to an unprecedented surge in data generation, posing significant challenges for large-scale ML tasks due to the substantial computational and communication resources required.

A practical solution is to adopt DML, which eliminates the need to collect or transfer massive volumes of data from edge devices—such as mobile phones—to a central server. Instead, model updates are exchanged while keeping data local, thereby reducing privacy risks and communication burdens.

The primary focus of this research is the design and evaluation of DML algorithms specifically tailored for wireless communication environments. Unlike conventional DML approaches that predominantly rely on first-order (gradient-based) optimization, this work leverages Hessian-based second-order methods to accelerate convergence, reduce the number of communication rounds, and improve robustness under realistic wireless conditions.

1.1 Background

Traditional ML paradigms face formidable challenges in the era of unprecedented data generation and escalating computational demands. The relentless growth in data volume, coupled with the need for advanced models, has led to an imperative shift towards more scalable and privacy-conscious approaches.

In many industries, data exists in isolated silos, separated from one another. This fragmentation arises from several factors, including competition between organizations, concerns over data privacy and security, and complex regulatory requirements. Even within a single organization, consolidating data from different departments can be challenging. Barriers include navigating stringent privacy regulations and overcoming techni-

cal or organizational boundaries that limit the integration of disparate datasets.

Assembling data distributed across multiple locations and organizational units is often complex and, in some cases, prohibitively expensive. Strict regulations governing data handling, privacy, and security further exacerbate these difficulties. The conventional approach of aggregating large datasets in a central repository for model training not only places significant strain on computational resources but also raises serious concerns regarding privacy and regulatory compliance. These challenges highlight the need for innovative approaches that enable effective collaboration across data silos. In this context, the FL paradigm emerges as a transformative strategy, redefining the landscape of machine learning.

This study is primarily segmented into two components: the first segment focuses on FL, while the latter part explores FDL. In FL, data is dispersed across multiple nodes (or edge devices, used interchangeably from here on), and an edge server facilitates communication among these nodes. The edge server plays a pivotal role in the learning process by receiving, aggregating, and broadcasting information from and to the nodes. In the original version of FL as introduced in the FedAvg algorithm McMahan et al. [2017a], the local models at each node are updated using the Stochastic Gradient Descent (SGD) optimization method. There have been numerous studies in the direction of FL using SGD, e.g., Zhang and Tao [2021], Amiri and Gündüz [2020a], Wang et al. [2019], Amiri and Gündüz [2020b]. However, the gradient-based optimization is highly suitable for the FL setup as local gradients average up to give an estimated global gradient; it is highly iterative and has high communication costs. This generates the need for Hessian-based optimization methods to reduce the iterations required for convergence and lower communication costs. This study explores methods to apply Hessian-based optimization in an FL setup. In the latter half of the study, we looked at Hessian-based FDL that operates without utilizing an edge server in the learning process. In this approach, nodes communicate directly, employing consensus algorithms. We design an algorithm called Network-GIANT with a combination of gradient-tracking and Newton-type iterations for each node. We further improved Network-GIANT's convergence performance to match GIANT's performance in our algorithm called Network Exact Consensus-GIANT (NEC-GIANT).

1.2 Federated Learning

The field of FL investigates the possibility of DML through collaboration directly from edge devices to enjoy the benefits of better privacy with the help of a central server. The concept was first introduced in 2017 McMahan et al. [2017a]. The authors define FL as follows:

Federated Learning is a machine learning setting where multiple entities collaborate

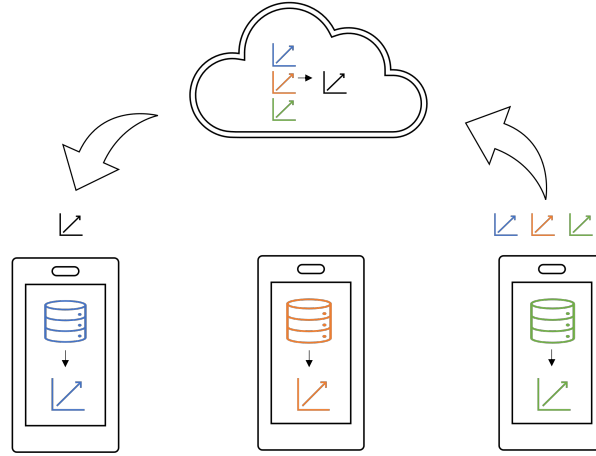


Figure 1.1: FL

in solving a machine learning problem under the coordination of a central server or service provider. Each entity's raw data is stored locally and not exchanged or transferred; instead, focused updates intended for immediate aggregation are used to achieve the learning objective

It is important to note that the focused updates derived from each entity are designed to carry the minimum information essential for the specific learning task.

The objective function in FL is given as,

$$F(\theta) = \frac{1}{N} \sum_{n=1}^N f_n(\theta)$$

where $f_n(\theta)$ is the local loss function $\ell(\mathbf{x}_n, y_n | \theta)$ for some parameter vector θ . The number of clients or nodes is represented by N . The aim is to minimize $F(\theta)$ over θ .

The process of FL has the following steps:

- **Initialization:** starting from an initial parameter vector θ , typically $\mathbf{0}$,
- **Local Evaluation:** perform local parameter vector updates at individual nodes, typically using gradient descent,
- **Updates Sharing:** sharing local updated parameter vectors from nodes to server,
- **Aggregation:** the server aggregates the received parameter vectors,
- **Broadcast:** the server broadcasts the aggregated parameter vector to the nodes,
- **Iteration:** iterate from **Local Evaluation** until convergence criteria are met.

The fields of ML and Artificial Intelligence (AI) have seen significant advancements in recent times. However, there are multiple domains where they have not been implemented to their fullest potential due to privacy and security concerns. FL, by its nature, provides a solution to this problem by extracting only the minimum essential information, thereby enhancing the privacy of the data source.

Beyond addressing privacy concerns, FL plays a crucial role in expediting the deployment and scalability of ML models. This is achieved by concurrently training multiple data points, reducing the necessity to transmit data to a central server. This becomes particularly advantageous in scenarios characterized by low bandwidth and high latency.

Initially introduced for mobile and edge device applications, FL has since expanded its applications into various domains, including healthcare and industrial engineering Li et al. [2020a], Yang et al. [2019]. Its adaptability and versatility make it a valuable tool not only for enhancing privacy but also for optimizing the efficiency of ML processes across diverse sectors. Some popular deployments of FL are in applications like Google's Gboard mobile keyboard Pichai [2019], doc.ai De Brouwer [2019] hot word detection in Snips Leroy et al. [2019].

1.2.1 Challenges

Despite its promising advantages, FL faces several challenges that need to be addressed for its widespread adoption and effectiveness. Some of the key challenges are as follows:

1. Communication Overhead

One of the primary challenges in FL is the transfer of model updates between the central server and edge devices. These updates can be high-dimensional, leading to significant bandwidth consumption. Without optimization, this can strain communication networks, especially in large-scale deployments. Thus, efficient communication protocols and model compression techniques are essential to reduce the overhead and maintain scalability Li et al. [2020b].

2. Device Heterogeneity

FL networks often involve a wide variety of edge devices with varying hardware specifications, network connectivity, and power availability. This heterogeneity leads to differences in storage and communication capabilities across clients. As a result, the effective participation rate of devices can fluctuate, impacting the performance and consistency of the global model Li et al. [2020b].

3. Limited Computational Resources

In many FL applications, such as IoT and remote sensing networks, devices have limited computational capacity. Although Hessian–vector products in second-order optimization can be computed efficiently without explicitly forming the Hessian, the additional computation and communication overhead can overwhelm resource-constrained edge devices, limiting the practicality of many Hessian-based fast-learning algorithms.

4. Privacy and Security Risks

Despite its decentralized nature, FL is not inherently immune to privacy and security threats. Although raw data remains on the user’s device, sensitive information can still be inferred from shared model parameters, gradients, or updates. Malicious participants, curious servers, or external attackers may exploit this vulnerability to extract information about other users’ data. Several types of inference attacks—such as membership inference, model inversion, and Generative Adversarial Network (GAN)-based reconstruction—have demonstrated the feasibility of such privacy breaches Mothukuri et al. [2021].

Additionally, malicious clients may attempt to poison the global model by submitting manipulated updates, either to corrupt the model’s behavior or to insert backdoors for future exploitation. These threats highlight the need for robust security and privacy-preserving mechanisms within the FL framework.

To address these privacy and security concerns, several mitigation strategies have been proposed and actively researched. The most prominent approaches include:

Differential Privacy (DP) Differential Privacy introduces statistical noise to model updates before they are shared with the server. This noise masks individual contributions, making it difficult for attackers to trace updates back to specific data points.

- Client-side DP is particularly effective, as noise is added before transmission, offering strong local privacy guarantees.
- However, excessive noise may reduce model accuracy, highlighting a trade-off between privacy and utility.
- DP has been effectively applied in federated learning algorithms such as NbAFL (Wei et al. [2020]) and Geyer et al. [2017] to enhance privacy protection, while maintaining acceptable levels of model performance..

Secure Multi-Party Computation (SMC) SMC enables multiple parties to compute a joint function (e.g., model aggregation) without revealing their inputs. In FL,

SMC ensures that no single party—including the server—can access raw updates from any client.

- It uses cryptographic techniques to aggregate encrypted updates, ensuring data confidentiality.
- While effective, SMC can introduce computational and communication overhead, making it less suitable for real-time or large-scale deployments unless optimized.

Homomorphic Encryption (HE) HE allows computations to be performed directly on encrypted data. In the FL context, clients can encrypt their model updates, and the server can still perform aggregation without decrypting them.

- This approach protects model updates end-to-end.
- However, HE is computationally intensive and requires careful engineering to be practical in real-world FL systems.

Anomaly and Behavior Detection To mitigate poisoning attacks, FL systems can incorporate robust aggregation algorithms and client behavior analysis.

- Techniques such as Krum, Trimmed Mean, and Median aggregation help identify and neutralize anomalous updates.
- Reputation-based systems and client auditing mechanisms can detect and exclude malicious clients from participation.

Adversarial Training GAN-based Defenses Some systems use adversarial training to build resilience against model inversion and evasion attacks. Additionally, GAN-based defenses can generate fake updates to confuse attackers, making it harder to isolate genuine data patterns.

1.2.2 Literature Review

The initial work in FL is in the direction of FederatedAveraging (McMahan et al. [2017a]), where the algorithm utilizes the local SGD on each node and an edge server to perform model averaging. The gradient-based DML can be easily parallelized due to the fact that the average of local gradients gives an unbiased estimate of the global gradient, and local gradients can be efficiently computed at the nodes. A major disadvantage of gradient-based DML methods is the high communication cost of sharing gradient updates due to multiple iterations for convergence and high communication bandwidth requirements. Thus, it is considered a fundamental bottleneck in DML Alistarh et al. [2017], Seide et al.

[2014], Lin et al. [2017]. Many studies performing gradient-based DML have emerged since the introduction of FederatedAveraging, especially targeting communication costs. This includes gradient compression - quantization, and sparsification - techniques. To reduce the communication rounds required to perform distributed learning tasks, multiple studies focus on second-order optimization methods, Zhang and Lin [2015], Shamir et al. [2014], Wang et al. [2018], Xu et al. [2020], Reddi et al. [2016], Jaggi et al. [2014], Gupta et al. [2021], resulting in faster convergence and hence reducing overall communication overhead.

First-Order Optimization Methods

Gradient Quantization The parallel SGD using 1-bit Quantization was implemented in Seide et al. [2014] and empirically showed convergence under certain assumptions. The authors in Alistarh et al. [2017] used a more flexible approach called the Quantized SGD (QSGD), a family of compression schemes where the number of bits sent by the nodes can be adjusted, giving the user an option to choose between accuracy and gradient precision. Wen et al. [2017] introduced the TernGrad, which uses 3-level gradients to accelerate distributed deep learning in data parallelism. Their approach can significantly reduce the communication time as it only requires three numerical levels, $\{-1,0,1\}$. Further reductions in the communication cost were achieved by Lin et al. [2017] through Deep Gradient Compression (DGC) using four methods: momentum correction, local gradient clipping, momentum factor masking, and warm-up training.

Gradient Sparsification In Gradient Sparsification, a sparse vector with a subset of essential values from the full gradient is shared among the nodes. Ström [2015] applied the gradient sparsification by sending gradient components larger than a predefined threshold. The method was proven to reduce communication bandwidth by three times in magnitude. Aji and Heafield [2017] implemented the gradient sparsification by setting the smallest 99% of updates (in terms of absolute value) to zero and then exchanging the resulting sparse vectors. This method resulted in up to 49% speed upgrade on MNIST without the loss of accuracy. Chen et al. [2018] and Renggli et al. [2019] empirically demonstrated the effectiveness of the Top-k sparsification in the SGD, showing a minimal impact on model convergence.

Jiang and Agrawal [2018] applied both the sparsification and the quantization techniques for the distributed SGD to solve a non-convex optimization problem.

Second-Order Optimization Methods

In distributed optimization, two key factors come into play: the computational power available at each machine and the communication overhead involved in the coordination between these machines. Using Newton-type methods, various studies reduce the convergence iterations, thereby reducing the high communication cost. Initial work includes the DANE (Shamir et al. [2014]), suitable for any smooth and strongly convex problem using an approximate Newton-type method. The DiSCO (Zhang and Lin [2015]) considers empirical risk minimization problems using the inexact damped Newton method. An approximate solution to the Newton step is obtained using a distributed preconditioned conjugate gradient method. Reddi et al. [2016] introduced the AIDE algorithm that improves convergence performance over the DANE using a simple first-order oracle. In the study Jaggi et al. [2014], a communication-efficient framework for distributed dual coordinate ascent algorithms, CoCoA, was presented to solve large-scale regularized loss minimization problems. GIANT (Wang et al. [2018]) is another popular second-order distributed algorithm that uses the conjugate gradient method for approximating the Newton direction and uses a globally improved gradient estimate. In the study Gupta et al. [2021] presented the LocalNewton algorithm that adaptively performs L number of local Newton iterations before aggregating the global Newton direction. When $L=1$, the LocalNewton behaves like the GIANT.

Over-the-air analog communication

Regarding wireless edge devices, reliably sending gathered data to a central processor might prove too energy and bandwidth-intensive, introduce significant delays, and potentially violate user constraints. Because wireless networks are commonly used for extensive data gathering and learning, numerous studies explore a wireless MAC connecting devices to the edge server. Over-the-air wireless transmission enhances communication efficiency and reduces bandwidth needs compared to traditional methods, which handle communication and computation separately. This improvement is made possible by leveraging the natural overlap of signals (superposition) in a wireless MAC. Doing so allows for the computation of functions like the average directly from the combined signals of concurrently transmitting devices. Amiri and Gündüz [2020a] demonstrated that, for distributed learning applications, over-the-air analog transmission from nodes to the parameter server significantly outperforms the digital transmission approach. The study in Yang et al. [2020] suggests employing a joint approach for device selection and receiver beamforming design to maximize the number of selected devices meeting the mean-squared-error (MSE) requirement for rapid model aggregation via over-the-air computation. In Zhu et al. [2019], the authors propose analog aggregation of simultaneously transmitted updates

by edge devices using over-the-air and exploiting the waveform-superposition property of the MAC. They proved that latency reduction with over-the-air multiple access with respect to the traditional orthogonal access scales linearly with the device population. A comparison between digital and analog communication for wireless distributed SGD is presented in Amiri and Gündüz [2020a]. The study demonstrates that over-the-air computation can significantly accelerate the learning process, particularly in low-power and bandwidth-constrained environments. Amiri and Gündüz [2019] proposes gradient sparsification, error accumulation, and compressive sensing to reduce the size and dimension of the parameter vector for limited channel bandwidth analog transmission in distributed SGD. The study also showed a power allocation scheme with average power constraints to cancel out the fading effect of the channel gain. Sery and Cohen [2020] worked on gradient-based distributed learning over noisy fading multiple access channels. The proposed algorithm, GBMA, works without eliminating the channel gain effect and directly uses noisy, distorted gradient updates. In Liu and Simeone [2020], the authors investigate adaptive power control for distributed gradient descent in wireless FL under privacy and power constraints. The benefits of over-the-air computing over orthogonal multiple access were demonstrated. The study reveals that utilizing channel noise as a privacy-inducing mechanism, measured through differential privacy, provides inherent privacy in wireless systems via uncoded transmission.

Other Methods

One of the major challenges in FL is to perform learning tasks in scenarios with variable communication capabilities and non-i.i.d. data distributions. FedProx (Li et al. [2020c]) is one such algorithm that builds upon FedAvg by allowing devices to perform varying amounts of training locally. This is determined by the device's available resources. The partially trained models from slower devices are then aggregated with the updates from faster devices to achieve the final model update, thereby targeting challenges due to system heterogeneity. Dal Fabbro et al. [2022] presented the SHED algorithm specifically targeting the challenges of heterogeneous communication resources. The SHED algorithm shares approximate local Hessian information using eigenvalue-eigenvector pairs, and the number of eigenvalue-eigenvector pairs shared varies as per the communication resource available at the node.

1.3 Fully Distributed Learning

While the central server plays an important role in supporting communication between the nodes in FL, it is not present in FDL. The nodes communicate among themselves to exchange each other's updates to achieve the learning process. The communication is

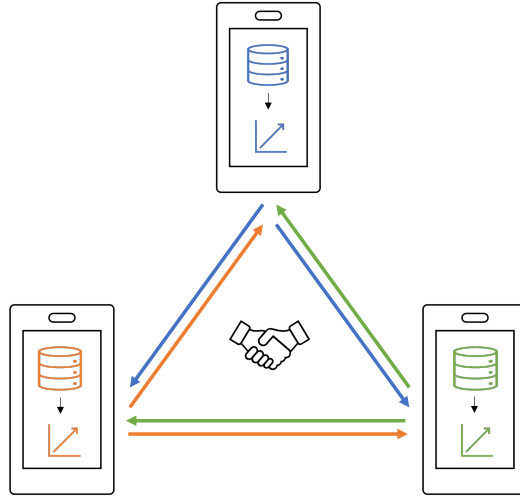


Figure 1.2: Fully Distributed Learning

illustrated through a network of N nodes, where edges denote the communication channel linking pairs of nodes. To achieve the learning process, the nodes perform consensus steps to aggregate local updates in each iteration.

The objective function in FDL remains consistent with that in FL. The learning process is as follows

- **Initialization:** starting with an initial parameter vector θ value, typically $\mathbf{0}$ vector,
- **Local Evaluation:** perform local parameter vector updates at individual nodes, typically using gradient descent,
- **Consensus:** perform local updates sharing among other nodes using consensus,
- **Iterate:** iterate from **Local Evaluation** until convergence criteria are met.

While the foundational principles guiding the FDL differ from those underpinning FL, they often find application in comparable problem domains. Shared challenges surface in both methodologies, fostering substantial convergence within the research communities involved.

1.3.1 Challenges

In FDL, training presents several challenges. Achieving global consistency across all nodes and efficiently finding the best solution is difficult. Information travels slowly through

many connections, potentially causing inconsistencies. This limited communication can significantly slow down the training and can significantly increase the communication cost. Computational Complexity at local nodes is another major challenge in FDL, as each node must perform its share of training with potentially limited memory and processing power compared to a centralized setting. While methods like second-order optimization could speed up training, they're not ideal for FDL. This is because calculating and sharing the Hessian is expensive and requires a lot of communication between nodes. Also, inverting the Hessian is difficult in a distributed setting. Even though these methods might make training faster, the extra communication might slow things down. Finding a balance between using more information and keeping communication efficient is a major challenge for training models.

Additionally, the system must be robust to node failures or network partitions, ensuring that the learning process can continue even if some nodes become unavailable.

1.3.2 Literature Review

Network Optimization

The initial work on the analysis of distributed computation models began back in the 1980s Bertsekas and Tsitsiklis [2015], Tsitsiklis [1984], Tsitsiklis et al. [1986]. With recent advancements and applications in ML, power systems, sensor networks, etc., distributed learning over networks has gained renewed interest. A decentralized computational framework for minimizing a composite objective function was investigated by Nedic and Ozdaglar [2009]. The objective function constitutes the summation of convex functions associated with multiple nodes over a time-varying topology. Jakovetić et al. [2014] presented fast distributed gradient algorithms for nodes in a network to tackle the distributed optimization problem of minimizing the sum of convex cost functions. Drawing inspiration from Nesterov's accelerated gradient method and consensus techniques, their algorithms demonstrated faster convergence rates compared to other gradient-based distributed algorithms available at that time. Applying ADMM to a general decentralised optimization problem, Shi et al. [2014] achieved global linear convergence assuming strong convexity of the cost function. The study Shi et al. [2015a] proposed the EXTRA algorithm for performing distributed consensus optimization across a network of nodes. This first-order optimization technique employs a fixed step size, deviating from the diminishing step sizes typically used in other distributed stochastic gradient methods. Instead of relying solely on the gradient of the last iteration, the EXTRA algorithm leverages the gradients of the past two iterations. This approach enables the algorithm to achieve faster convergence rates compared to its contemporary first-order counterparts.

To achieve even faster results and reduce communication costs, approximate Newton

methods in fully distributed settings have been studied in the literature. Zargham et al. [2013] examined dual descent algorithms that approximate the Newton direction, aiming to provide rapid distributed solutions for the convex network flow optimization problem. The proposed algorithm ADD exhibits super-linear convergence within the neighborhood of the optimal value using the sparse Taylor approximation of the inverse Hessian. An alternative approach utilizing Taylor series expansion to approximate the Newton step was introduced by Mokhtari et al. [2016a]. Their proposed Network Newton-K (NN-K) algorithm incorporates K terms from the Taylor series expansion to estimate the Newton step. This method demonstrates at least linear convergence to a near-optimal solution while also highlighting a trade-off between convergence speed and proximity to the optimal solution. Network-DANE is another Newton-type FDL algorithm that transforms the well-known DANE (Shamir et al. [2014]) algorithm for the FL framework to the Network Learning framework by employing gradient tracking. A global superlinear convergence for FDL was achieved by Zhang et al. [2022]. The study introduced an algorithm called DAN-LA, which integrates two key components: a finite-time set consensus method and Polyak’s adaptive step size technique. A novel Newton tracking approach was investigated by Zhang et al. [2021] that combines second-order information with gradient tracking to enhance the convergence speed. The proposed Newton tracking algorithm demonstrates a linear convergence rate to the exact optimal solution under the assumption of strong convexity.

1.4 Research Objectives

1.4.1 Resource-Efficient Communication

The communication overhead has been one of the main challenges in FL. Most existing studies on DML have focused on traditional FDM/TDM communication schemes where the bandwidth requirement is proportional to the number of nodes in the network. We aim to address this challenge of communication in DML using a wireless fading MAC. In this scenario, information from each distributed node is transmitted as an analog signal through a noisy fading wireless MAC using a common shaping waveform. The edge server receives a superposition of these analog signals, computes a new parameter estimate, and communicates it back to the nodes. This iterative process continues until an appropriate convergence criterion is met.

1.4.2 Fast Convergence with lower Computational complexity

Typical FL approaches rely on the communication of local gradients and averaging at the edge server. This approach requires a high number of communication rounds due to a slow

convergence rate. This study investigates faster convergence by implementing a second-order optimization using Determinantal Averaging. To alleviate the computational load at each node, the study explores sub-sampled Newton methods to compute approximate local Hessian and “lazy”^a methods where the Hessian is computed only if necessary.

1.4.3 Node selection and Power control

In real-world communication scenarios, nodes encounter varying channel fading conditions, impacting signal quality. Enhancing the signal-to-noise ratio is necessary to mitigate this issue, requiring increased power consumption at the nodes. Our objective is to address this challenge by opting to involve only a selected subset of nodes in the learning process based on a threshold set for the channel gain effect. Additionally, we incorporate a channel inversion strategy to mitigate the impact of channel gain on signal quality.

1.4.4 Improved Convergence in Fully Distributed Learning

Our objective is to tackle the challenge of distributed multi-agent learning, where the primary goal is to minimize the cumulative sum of local objective functions, referred to as empirical loss functions, through local optimization and information exchange among neighboring nodes, independently of a central server.

The study introduces a novel Newton-type fully distributed optimization algorithm named Network-GIANT. This algorithm is built upon the framework of GIANT, an FL algorithm that relies on a central parameter server. Network-GIANT combines gradient-tracking and a Newton-type iterative algorithm at each node, with consensus-based averaging of Newton updates.

The primary objective of Network-GIANT is to achieve efficiency in terms of both communication cost and run time, making it particularly suitable for wireless networks.

1.5 Thesis Contributions

This thesis addresses critical challenges in both FL and FDL environments. Our primary focus is on reducing communication costs during the learning process through the application of second-order optimization methods.

In the FL domain, we introduce Distributed Approximate Hessian with Determinantal Averaging Algorithm (DANDA), a Newton-type algorithm that substantially decreases the number of communication rounds needed for convergence. Acknowledging the computational constraints of local nodes, we incorporate approximation methods for Hessian

^aWe refer to this approach as a Lazy Hessian update, although in strict terms it corresponds to a conditional Hessian update, since the Hessian is recomputed only when the gradient change between consecutive iterations exceeds a threshold.

calculation. DANDA utilizes over-the-air MAC for communication, and we provide experimental results demonstrating the algorithm’s performance under channel fading and noise conditions.

We advance the field of DML over wireless networks by proposing a novel approximate Newton-based algorithm that aims to minimize communication and computation costs in wireless fading MAC environments. Our method employs analog signal transmission with a unified waveshaping waveform, adapting to noisy and fading conditions, and incorporates a channel gain threshold-based node selection mechanism. We introduce an approximate Newton approach utilizing sub-sampled Hessians and weighted averaging of local Hessians, combined with an adaptive Hessian update strategy for improved computational efficiency. This leads to two new algorithms: Lazy-DANDA and Lazy-DANTA. The adaptive strategy further reduces communication overhead by sharing local Hessian updates only when necessary. To address variable channel fading effects on node signals, which can significantly distort the aggregated signal at the central server, we implement channel inversion and power control techniques. These measures help regulate power consumption and maintain the quality of received signals from the nodes.

In the fully distributed scenario, we extend GIANT, a FL algorithm dependent on a central server, to a scenario where nodes communicate with their neighboring nodes only, without the presence of a central server, by incorporating gradient-tracking and Newton-type iterative methods at each node alongside consensus-based averaging of Newton updates. Network-GIANT is designed to be efficient in both communication cost and run-time, making it particularly suitable for wireless networks. The algorithm ensures semi-global and exponential convergence to the exact solution, assuming the local loss functions are strongly convex and smooth. The algorithm explicitly targets the slow convergence challenge of FDL by incorporating second-order optimization, thereby accelerating the learning process. We further improved Network-GIANT to show convergence properties similar to GIANT in the FL setup. We call this algorithm Network Exact Convergence-GIANT. This algorithm uses finite-time distributed consensus to build exact consensus among the nodes, efficiently performing the learning task.

1.6 Thesis outline

The rest of the thesis is organized into four parts.

Chapter 2 presents Distributed Approximate Hessian with Determinantal Averaging Algorithm (DANDA). The algorithm performs Newton-based optimization using determinantal averaging for over-the-air distributed learning. The algorithm proves to have the benefit of fast convergence with reduced computational cost per iteration. The results section compares the DANDA with other FL algorithms.

Chapter 3 presents further improvements in the DANDA algorithm in terms of its energy consumption and accuracy as we incorporate power control and channel inversion techniques. The power control is done using a power adjustment factor, which checks the power consumption per iteration. In channel inversion, the nodes transmit information only when the channel gain is higher than the channel gain threshold. We also demonstrate a “lazy” version of the algorithm that makes the algorithm computationally efficient by calculating the Newton direction only when required. The theoretical analysis shows a super-linear convergence of the algorithm. Empirical results demonstrate that the algorithm performs well in comparison with other cutting-edge distributed learning algorithms.

Chapter 4 presents the FDL scenario. We implemented a decentralized version of the GIANT algorithm called Network-GIANT. The Network-GIANT is designed via a combination of gradient-tracking and a Newton-type iterative algorithm at each node, with consensus-based averaging of Newton updates. We implement an exact average consensus technique called “Finite-Time Distributed Consensus” and match the convergence performance of GIANT in a separate algorithm called NEC-GIANT. We show theoretical analysis and empirical results to evaluate convergence performance.

Finally, Chapter 5 concludes the thesis and comments about the direction of future work.

2

Distributed Approximate Hessian with Determinantal Averaging

2.1 Introduction

Privacy concerns often restrict the applications of ML in real-world challenges. Conventionally, ML tasks are performed on a server that contains data gathered from multiple edge devices or nodes (e.g., mobiles, sensors). It is difficult to preserve privacy as the user data has to be transferred from these nodes to the server for model training in “centralized” learning tasks. DML provides a method to train the learning model locally without collecting data from the nodes. An edge server that coordinates among the nodes is often used in DML. While training, the nodes share training parameters with the edge server, which performs aggregation and provides a new parameter estimate back to the nodes. This process continues until the training parameters attain a global optimum value within a certain tolerance. This way of learning discards the need to move end-user data for model training, thereby securing privacy as well as reducing communication overhead.

As nodes exchange training parameters with the edge server, there is an ongoing requirement for communication. This becomes particularly problematic in environments with limited bandwidth or high latency, which can hinder the training process and diminish the system’s overall efficiency. Furthermore, the frequency of communication between the nodes and the edge server can influence the convergence rate of the training algorithm.

2.1.1 Related Work

This work is related to the field of FL (McMahan et al. [2017a]), where the nodes contribute to the learning process with the help of an edge server. The nodes exchange local updates based on the local dataset with the edge server and perform the learning task, typically using the SGD technique. Conventional FL algorithms use an orthogonal access channel to communicate information from the nodes, which exhausts the bandwidth as the

number of nodes N increases. Numerous studies show the application of distributed-SGD over wireless MAC using an analog uncoded transmission, utilizing over-the-air additive nature of the wireless channel, and reducing the channel bandwidth requirement, e.g., Amiri and Gündüz [2020b], Zhu et al. [2019], Amiri and Gündüz [2019], Sery and Cohen [2020], Guo et al. [2020], Sery et al. [2021]. The works of Amiri and Gündüz [2020b], Zhu et al. [2019], Amiri and Gündüz [2019] use analog MAC transmission methods with scheduling schemes and power control to compensate for the wireless fading channels. The GBMA algorithm in Sery and Cohen [2020] works directly with distorted gradients transmitted over the MAC using analog transmission. In contrast, Guo et al. [2020] proposes the optimization of the parameters in the transceiver with consideration of the “non-stationarity” in the local gradients based on a simple feedback variable. In Sery et al. [2021], the authors propose pre-coding at the nodes and scaling at the parameter server to mitigate the effect of channel noise.

The iterative algorithms based on the transmission of local gradients have a significant communication overhead due to their slow convergence (i.e., many communication rounds). Second-order optimization-based methods ADMM used at the local nodes to alleviate the communication cost, although they are computationally expensive. Extensive literature on reducing the communication overhead by using quasi-Newton methods is present, including the DANE (Shamir et al. [2014]), which approximates the Newton step in a distributed manner using Bregman divergence, DONE (Dinh et al. [2022]), which produces an approximate Newton direction using classical Richardson iteration on each edge node. Other methods include GIANT (Wang et al. [2018]) and DISCO (Zhang and Lin [2015]) that approximate the Hessian using the conjugate gradient method. One method of reducing the communication overhead is using the Newton method locally, and instead of sharing Newton direction for global parameter vector update, optimize the parameter vector locally and share the updated parameters as shown in the algorithm LocalNewton (LN) Gupta et al. [2021]. Hua et al. [2019] compute Newton’s direction locally and share the local optimum parameter vector to the edge server by adopting a nonconvex low-rank beamforming approach with over-the-air computation via difference-of-convex-functions programming. Most second-order Newton-type methods, however, require computation of the inverse of the Hessians locally, which is computationally expensive, thus warranting various forms of approximate Newton methods. In a recent work Krouka et al. [2022], the authors determined the inverse Hessian times the gradient product vector using the ADMM to reduce computation cost.

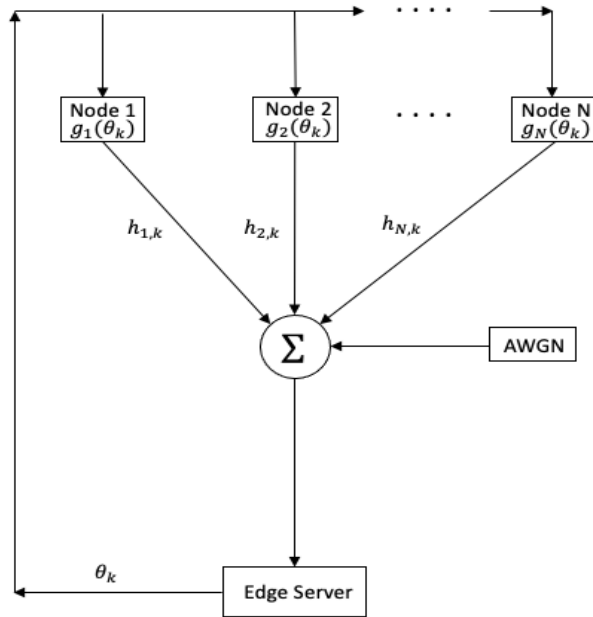


Figure 2.1: Transmission Scheme over a wireless MAC

2.1.2 Our Contributions

In this study, we introduce an innovative DML approach founded on a computationally efficient approximate Newton method, utilizing the determinantal averaging technique formulated by Derezhinski and Mahoney [2019] at each node. The nodes communicate training parameters with the edge server using analog uncoded transmission over a fading wireless MAC, with a communication overhead of $O(d)$ at each iteration, where d is the dimension of the parameter vector θ to be optimized. Based on the above concept, we present a DML algorithm, titled Distributed Approximate Newton with Determinantal Averaging (DANDA), and show its empirical tests on a linear regression task with the Million Song Dataset Bertin-Mahieux et al. [2011], in a simulated communication setting with its performance averaged over multiple random channel realizations. Comprehensive numerical results illustrate significantly faster convergence of DANDA compared to first-order methods such as GBMA and comparable energy consumption and computational complexity with state-of-the-art second-order methods.

The subsequent sections of the chapter are structured as follows. Section 2.2 presents a DML and communication model to establish a foundational understanding of the concepts involved. Section 2.3 offers the theoretical underpinnings of the DANDA algorithm, along with insights into its implementation. Following the theoretical exposition, Section 2.4 furnishes a detailed compilation of experimental results that delineate the comparative

performance of DANDA against existing algorithms. Finally, Section 2.5 encapsulates the chapter with concluding remarks, synthesizing the key findings and implications derived from the study.

2.2 System Model

2.2.1 Distributed Learning Model

We consider a distributed wireless edge learning problem with N nodes and an edge server for aggregating information collected from the nodes over a fading wireless MAC. The global objective function, $F(\boldsymbol{\theta})$, which is an average of the local objective functions, $f_n(\boldsymbol{\theta})$, can be described as follows:

$$F(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N f_n(\boldsymbol{\theta}), \quad (2.1)$$

$$f_n(\boldsymbol{\theta}) = \frac{1}{b_n} \sum_{i=1}^{b_n} \ell_n(\boldsymbol{\theta}^T \mathbf{x}_{i,n})$$

where $\boldsymbol{\theta}$ is a $d \times 1$ parameter vector over which the optimization of the global objective function, $F(\boldsymbol{\theta})$ takes place, ℓ_n is the loss function at the n -th node, $\mathbf{x}_{i,n}$ is the i^{th} input row for n^{th} node and b_n is the size of the local dataset in the n^{th} node. As shown in Fig. 1, the edge server broadcasts an initial parameter vector, $\boldsymbol{\theta}_0$, to all the nodes, to which each node determines a function of the local information on its data. These results are then communicated to the edge server over the wireless MAC as analog signals with a common shaping waveform, and due to the superposition property of the wireless medium, the transmitted signals get aggregated over the transmission channel and consequently provide the global descent direction at the edge server.

2.2.2 Communication Model

We consider a wireless MAC, where each node transmits information as a common analog waveform, and the superposition of the N transmitted waveforms is received at the edge server as shown in Fig. 2.1. In the situation where the number of features, d , is much smaller than the number of nodes, N , this communication scheme offers a more effective solution to the issue of high bandwidth requirement compared to traditional orthogonal multiple access, as communication overhead is proportional to d instead of N .

Assuming time slots, with slot duration T , with the k -th slot being $t_k \leq t \leq t_k + T$. Let $\mathbf{s}(t) = (s_1(t), s_2(t), \dots, s_d(t))$, $0 < t < T$, be a vector of d orthogonal base-band equivalent normalized waveforms, satisfying $\int_0^T s_m^2(t) dt = 1$, $\int_0^T s_m(t) s_i(t) dt = 0$, for $m \neq n$. Each node n experiences at the k -th slot a block fading channel $\tilde{h}_{n,k}$ with gain $h_{n,k} \triangleq |\tilde{h}_{n,k}| \in \mathbb{R}_+$

2.3. DISTRIBUTED NEWTON ALGORITHMS WITH DETERMINANTAL AVERAGING

and phase $\phi_{n,k} \triangleq \angle \tilde{h}_{n,k} \in \{-\pi, \pi\}$. The channel fading is assumed independent and identically distributed (*i.i.d.* across the nodes), and time slots, with mean μ_h and variance σ_h^2 . We also assume that each iteration of the DML algorithm is aligned with the time slots. In other words, the fading channel coefficients are constant within a single iteration but change from iteration to iteration in a statistically independent manner.

NOTE: The phase lag due to the channel experienced by each node $\phi_{n,k}$ is assumed to be known at each node and, therefore, canceled before transmitting a signal. This is the underlying principle of transmission over a coherent MAC, which requires distributed transmit beamforming and can be difficult to achieve across many nodes. Note, however, that even if the channel phases are not accurately accounted for, and there is some residual phase offset, the individual signals arriving at the edge server will still have positive coefficients as long as the phase offsets are small Sery and Cohen [2020]. A description of valuable notations is provided in Table 1.

2.3 Distributed Newton algorithms with Determinantal Averaging

We aim to solve the empirical risk minimization problem described in (2.1) where $f_n(\cdot) : \mathbb{R}^d \mapsto \mathbb{R}$, for all $n \in [N] = \{1, 2, \dots, N\}$. The function $f_n(\cdot)$ satisfies the following assumptions:

Assumption 2.1. $f_n(\cdot)$ is Convex and twice differentiable.,

$$\nabla^2 f_n(\cdot) \succcurlyeq 0$$

Assumption 2.2. $f_n(\cdot)$ has a Lipschitz continuous Hessian: For any $n \in [N]$, there exists a constant L , such that $\forall \boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d$,

$$\|\mathbf{H}_n(\boldsymbol{\theta}) - \mathbf{H}_n(\boldsymbol{\theta}')\|_2 \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$$

where, $\mathbf{H}_n(\boldsymbol{\theta}) = \nabla^2 f_n(\boldsymbol{\theta})$

Note: The assumptions mentioned above are standard in the ML literature and are satisfied by several standard cost functions, including the linear regression-based least squares cost function.

2.3.1 Distributed Newton with Determinantal Averaging: DNDA

In the distributed setting, we have N nodes and each node carries a subset $\mathcal{S}_n \subset \mathcal{R}$ of size b_n (i.e., $b_n = |\mathcal{S}_n|$), for all $n \in [N] = \{1, 2, \dots, N\}$, where \mathcal{R} is the original dataset. The

2.3. DISTRIBUTED NEWTON ALGORITHMS WITH DETERMINANTAL AVERAGING

samples \mathcal{S}_n at each node are obtained randomly without replacement, hence $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset, \forall i, j \in [N], i \neq j$.

Note that a typical second-order optimization algorithm based on approximating the global Newton step, $\mathbf{H}^{-1}(\boldsymbol{\theta})\mathbf{G}(\boldsymbol{\theta})$ for $F(\boldsymbol{\theta})$, involves aggregating the local Hessians and the local gradients from the LN steps, $\mathbf{H}_n^{-1}(\boldsymbol{\theta})\mathbf{g}_n(\boldsymbol{\theta})$ for $f_n(\boldsymbol{\theta})$, from the nodes. However, obtaining the Hessian for the global Newton's step by naive aggregation of the LN steps suffers an inversion bias as $\mathbb{E}(\mathbf{H}_n^{-1}(\boldsymbol{\theta}_k)) \neq \mathbf{H}^{-1}(\boldsymbol{\theta}_k)$. Derezinski *et. al.* in Derezinski and Mahoney [2019] proved that a weighted sum of local inverse Hessian times the global gradient asymptotically converges to global inverse Hessian times the global gradient. They call this Determinantal Averaging, as given below. Define

$$\mathbf{H}^{-1}(\widehat{\boldsymbol{\theta}})\widehat{\mathbf{G}}(\boldsymbol{\theta}) = \frac{\sum_{n=1}^N a_n \mathbf{H}_n^{-1}(\boldsymbol{\theta})\mathbf{G}(\boldsymbol{\theta})}{\sum_{n=1}^N a_n} \quad (2.2)$$

where $a_n = \det(\mathbf{H}_n(\boldsymbol{\theta}))$ and $\mathbf{G}(\boldsymbol{\theta})$ is the global gradient. As $N \rightarrow \infty$, $\mathbf{H}^{-1}(\widehat{\boldsymbol{\theta}})\widehat{\mathbf{G}}(\boldsymbol{\theta}) \rightarrow \mathbf{H}^{-1}(\boldsymbol{\theta})\mathbf{G}(\boldsymbol{\theta})$ almost surely. Note that this estimation requires the product of local inverse Hessian and the global gradient,^a which is a vector of size $d \times 1$. As the number of nodes, $N \rightarrow \infty$, the estimate of inverse of the global Hessian $\widehat{\mathbf{H}}^{-1} \rightarrow \mathbf{H}^{-1}$. The convergence guarantee offered by this method is given by:

Corollary. *Derezinski and Mahoney [2019]: For any $\delta, \eta \in (0, 1)$ if the expected local sample size satisfies $b_n \geq C\eta^{-2}\mu d^2 \log^3 \frac{d}{\delta}$ then under Assumption 2.2*

$$\|\tilde{\boldsymbol{\theta}} - \boldsymbol{\theta}^*\| \leq \max \left\{ \frac{\eta}{\sqrt{N}} \sqrt{K} \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|, \frac{2L}{\lambda_{\min}} \|\boldsymbol{\theta} - \boldsymbol{\theta}^*\|^2 \right\}$$

where $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} - \mathbf{H}^{-1}(\widehat{\boldsymbol{\theta}})\widehat{\mathbf{G}}(\boldsymbol{\theta})$,

holds with probability at least $1 - \delta$, where C is an absolute constant, b_n is the size of local dataset, λ_{\min} is the smallest eigenvalue of $\nabla^2 F(\boldsymbol{\theta})$ and $\mu = \frac{1}{d} \max_i \ell''(\boldsymbol{\theta}^T \mathbf{x}_i)(\mathbf{x}_i^T \nabla^{-2} F(\boldsymbol{\theta}) \mathbf{x}_i)$

It can be observed that as $N \rightarrow \infty$, the above bound exhibits quadratic convergence similar to exact Newton's method.

In our analog transmission-based DML algorithm, The signal to be transmitted by the n^{th} node at the k^{th} iteration is sent to the edge server in two halves of the time slot, representing functions of the numerator and the denominator in (2.2). The signals transmitted from the nodes in the first ($t_k \leq t < t_k + \frac{T}{2}$) and second half ($t_k + \frac{T}{2} \leq t <$

^aHowever, as shown in Krouka et al. [2022], the LN's step can be replaced by the product of local Hessian and local gradient, $(\mathbf{H}_n^{-1}(\boldsymbol{\theta})\mathbf{g}_n(\boldsymbol{\theta}))$, as long as $\mathbf{g}_n(\boldsymbol{\theta})$ is an i.i.d. unbiased estimate of the global gradient.

2.3. DISTRIBUTED NEWTON ALGORITHMS WITH DETERMINANTAL AVERAGING

$t_k + T$) of the k -th time slot, respectively, as

$$y_n^1(\boldsymbol{\theta}_k, t) \triangleq \sqrt{\alpha P} e^{-j\phi_{n,k}} a_n (\mathbf{H}_n^{-1}(\boldsymbol{\theta}_k) \mathbf{g}_n(\boldsymbol{\theta}_k))^T \mathbf{s}(t) \quad (2.3)$$

$$y_n^2(\boldsymbol{\theta}_k, t) \triangleq \sqrt{(1-\alpha)P} e^{-j\phi_{n,k}} a_n s'(t) \quad (2.4)$$

Here, $a_n = \det(\mathbf{H}_n(\boldsymbol{\theta}))$, $e^{-j\phi_{n,k}}$ is the phase correction factor to produce a positive channel gain at the receiver, $\mathbf{s}(t)$ is the vector of orthogonal normalised waveforms to carry the numerator part of (2.2) and $s'(t)$ is a scalar normalised waveform function to carry the denominator part of (2.2). P is a node power adjustment factor (PAF), which is distributed among the two signals such that αP is the power to the first signal representing the numerator and $(1-\alpha)P$ is the power to the denominator, where $\alpha \in (0, 1)$. Signals received at the edge server are superimposed on each other due to the analog nature of the transmitted signals. The two received signals in the first and second halves of the k -th time slot are, respectively:

$$\mathbf{r}_k^1(t) = \sum_{n=1}^N \sqrt{\alpha P} h_{n,k} a_n (\mathbf{H}_n^{-1}(\boldsymbol{\theta}_k) \mathbf{g}_n(\boldsymbol{\theta}_k))^T \mathbf{s}(t) + \mathbf{w}_k^1(t) \quad (2.5)$$

$$r_k^2(t) = \sum_{n=1}^N \sqrt{(1-\alpha)P} h_{n,k} a_n s'(t) + w_k^2(t) \quad (2.6)$$

Here \mathbf{w}_k^1 and w_k^2 are additive independent white Gaussian noise (AWGN) processes distributed as $\mathcal{N}(0, \sigma_w^2 \mathbf{I}_d)$ and $\mathcal{N}(0, \sigma_w^2)$ respectively. After matched filtering and some readjustments, the demodulated signals at the edge server are:

$$\mathbf{v}_k^1 = \frac{1}{N} \sum_{n=1}^N h_{n,k} a_n \mathbf{H}_n^{-1}(\boldsymbol{\theta}_k) \mathbf{g}_n(\boldsymbol{\theta}_k) + \mathbf{w}_k^1 \quad (2.7)$$

$$v_k^2 = \frac{1}{N} \sum_{n=1}^N h_{n,k} a_n + w_k^2 \quad (2.8)$$

where $\mathbf{w}_k^1 \sim \mathcal{N}(0, \frac{\sigma_w^2 \mathbf{I}_d}{N^2 \alpha P})$ and $w_k^2 \sim \mathcal{N}(0, \frac{\sigma_w^2}{N^2 (1-\alpha) P})$. The edge server receives the superposition of the respective signals from all the nodes with channel distortion and AWGN. The ratio of two signals, $\frac{\mathbf{v}_k^1}{v_k^2}$, gives the noisy and distorted version of Newton's step, which is used for parameter updates as follows,

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \beta \frac{\mathbf{v}_k^1}{v_k^2} \quad (2.9)$$

2.3. DISTRIBUTED NEWTON ALGORITHMS WITH DETERMINANTAL AVERAGING

Where β is an appropriately scaled learning rate.

Note that the communication overhead per node involves a $d+1$ dimensional vector at each iteration, requiring a similar communication overhead compared to gradient-based methods that require a d dimensional vector at each iteration. The difficulty, of course, lies in computing the inverse local Hessian at the nodes, which generally costs $O(d^3)$ computations per iteration. In Section 2.3.3, we will address this issue by using a quasi-Newton method to approximate the local Hessians.

2.3.2 DNDA with Recursive Least Squares

Consider the special case where the local cost functions are regularized least square loss functions (i.e., regularized linear regression), given as follows:

$$f_n(\boldsymbol{\theta}) = \ell_n(\boldsymbol{\theta}^T \mathbf{x}_n) = \frac{1}{2} \|\boldsymbol{\theta}^T \mathbf{x}_n - \mathbf{y}_n\|_2^2 + \frac{\lambda}{2} \|\boldsymbol{\theta}\|_2^2 \quad (2.10)$$

It is easy to check that in the case of regularized linear regression, the local Hessian is independent of the parameter vector $\boldsymbol{\theta}$ and hence is constant over the iterations of the DNDA algorithm. Therefore, the denominator of the (2.2), which involves the sum of the determinants of the local Hessian, can be estimated using a recursive least squares estimation method. For the k^{th} iteration, the received denominator of (2.2) is a distorted version of the determinant of the local Hessian, a_n summed over all the nodes, given by v_k^2 in (2.8).

By observing the received signal v_k^2 over \bar{T} iterations, We can estimate the local Hessian determinants, a_n at k^{th} iteration using an ordinary least square problem as follows:

$$\hat{\mathbf{a}}_k = \left(\sum_{k=1}^{\bar{T}} \mathbf{h}_k \mathbf{h}_k^T \right)^{-1} \left(\sum_{k=1}^{\bar{T}} \mathbf{h}_k z_k \right)$$

Where \mathbf{h}_k is a vector of channel gains observed by the nodes 1 to N for k^{th} iteration, $\hat{\mathbf{a}}_k$ is a vector of estimated local Hessian determinants for nodes 1 to N for k^{th} iteration and $z_k = N v_k^2$. A recursive least squares (RLS) method can be used to estimate \mathbf{a}_n iteratively as follows,

$$\begin{aligned} \hat{\mathbf{a}}_k &= \hat{\mathbf{a}}_{k-1} + \mathcal{K}_k (z_k - (\mathbf{h}_k)^T \hat{\mathbf{a}}_{k-1}) \\ \mathcal{K}_k &= \mathcal{P}_k \mathbf{h}_k, \quad \mathcal{P}_k = \frac{\mathcal{P}_{k-1}}{\mathbf{h}_k^T \mathcal{P}_{k-1} \mathbf{h}_k + 1} \end{aligned} \quad (2.11)$$

where, \mathcal{K}_k is the gain vector for the k^{th} iteration and \mathcal{P}_k represents $(\sum_{k=1}^{\bar{T}} \mathbf{h}_k \mathbf{h}_k^T)^{-1}$, computed in a recursive way using the matrix inversion lemma.

This method avoids the direct use of a distorted sum of local Hessian determinants,

2.3. DISTRIBUTED NEWTON ALGORITHMS WITH DETERMINANTAL AVERAGING

$\sum_n a_n$, and instead uses an estimate of it. Using a recursive least squares estimate of $\sum_n a_n$ provides better asymptotic convergence compared to the DNDA algorithm using the distorted sum of local Hessians, as shown in the Experimental Results section.

2.3.3 Distributed Approximate Newton with Determinantal Averaging: DANDA

Computing the inverse Hessian in Newton's step is expensive, especially for a high-dimensional dataset, and the computational constraints at the edge nodes may render it even more prohibitive. Here, we adopt an approximate Newton step using a sub-sampled Hessian, as presented by Erdogdu and Montanari in their algorithm called NewSamp (Erdogdu and Montanari [2015]). This algorithm has been shown to significantly reduce computational cost while preserving a second-order method's fast convergence rate. Other than the assumptions of convexity and differentiability of the local cost functions and Lipschitz continuity of the Hessian, i.e., 2.1 and 2.2, we need two additional assumptions to use the sub-sampled Hessian:

Assumption 2.3. *Bounded Hessian:* $\forall n = 1, 2, \dots, N$, the Hessian of the function $f_n(\boldsymbol{\theta})$, i.e., $\nabla^2 f_n(\boldsymbol{\theta})$, is upper bounded by an absolute constant M , i.e., $\max_{n \leq N} \|\nabla^2 f_n(\boldsymbol{\theta})\| \leq M$.

Assumption 2.4. *Lipschitz continuous sub-sampled Hessian:* For any subset, $S_n \subset [\mathcal{S}_n]$ in $n \in [N]$, there exists constant L_n , such that $\forall \boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d$,

$$\|\mathbf{H}_{S_n}(\boldsymbol{\theta}) - \mathbf{H}_{S_n}(\boldsymbol{\theta}')\|_2 \leq L_n \|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$$

where, $\mathbf{H}_{S_n}(\boldsymbol{\theta}) = \nabla^2 f_n(\boldsymbol{\theta})$ over the subset S_n .

The approach in Erdogdu and Montanari [2015] is based on a low-rank approximation of the sub-sampled Hessian, i.e., the Hessian obtained over the subset of data points of size S_n . The low-rank approximation of local sub-sampled Hessian $\mathbf{H}_{S_n}(\boldsymbol{\theta})$ is performed by using a truncated singular value decomposition (SVD) along the direction of large eigenvalues only, as, $\hat{\mathbf{H}}_{S_n^k} = \mathbf{U}_r^k (\boldsymbol{\Lambda}_r^k) \mathbf{U}_r^{kT}$, where $\boldsymbol{\Lambda}_r$ is $r \times r$ diagonal matrix with the r largest eigenvalues of the local sub-sampled Hessian $\mathbf{H}_{S_n^k}(\boldsymbol{\theta})$ and \mathbf{U}_r is a $d \times r$ matrix whose columns correspond to the eigenvectors of $\mathbf{H}_{S_n^k}(\boldsymbol{\theta})$. The superscript k represents the iteration number.

For stability, the eigenvalues smaller than r^{th} largest eigenvalue are replaced by $(r+1)^{th}$ largest eigenvalue of $\hat{\mathbf{H}}_n(\boldsymbol{\theta})$, and the approximate inverse Hessian is then given by

$$\hat{\mathbf{H}}_{S_n^k}^{-1}(\boldsymbol{\theta}) = \lambda_{r+1}^{k-1} \mathbf{I}_d + \mathbf{U}_r^k (\boldsymbol{\Lambda}_r^{k-1} - \lambda_{r+1}^{k-1} \mathbf{I}_r) \mathbf{U}_r^{kT} \quad (2.12)$$

The computational cost in classical second-order methods is $\mathcal{O}(b_n d^2)$ operations per node for the local Hessian and $\mathcal{O}(d^3)$ operations per node for the inverse of the local

Algorithm 1 DANDA

input: $f_n(\cdot)$; $\boldsymbol{\theta}_0 \in \mathbb{R}^d$; λ ; β ; P ; α^* ; S_n^k ; ϵ
define: $[\mathbf{U}_r, \boldsymbol{\Lambda}_r] = \text{TruncatedSVD}_r(\mathbf{H}_{S_n})$ as rank- r truncated SVD of local sub-sampled Hessian $(\mathbf{H}_{S_n}(\boldsymbol{\theta}))$ with $(\boldsymbol{\Lambda}_r)_{ii} = \lambda_i$;

 $k = 1$; $\boldsymbol{\theta}_k = \boldsymbol{\theta}_0$
while $\frac{|F(\boldsymbol{\theta}_k) - F(\boldsymbol{\theta}_{k+1})|}{F(\boldsymbol{\theta}_k)} \geq \epsilon$ **do**

Local training process

for $n = 1$ to N **do**

Compute local gradient

$$\mathbf{g}_n(\boldsymbol{\theta}) = \nabla f_n(\boldsymbol{\theta}_k)$$

Compute local Hessian using sub-samples

$$\mathbf{H}_{S_n^k}(\boldsymbol{\theta}) = \frac{1}{|S_n^k|} \sum_{i \in S_n^k} \ell''_n(\boldsymbol{\theta}_k^T \mathbf{x}_{i,n})$$

$$\hat{\mathbf{H}}_{S_n^k}^{-1}(\boldsymbol{\theta}) = \lambda_{r+1}^{k-1} \mathbf{I}_d + \mathbf{U}_r^k (\boldsymbol{\Lambda}_r^{k-1} - \lambda_{r+1}^{k-1} \mathbf{I}_r) \mathbf{U}_r^{kT}$$

 Compute the numerator \mathbf{N}_n^k and the denominator \mathbf{D}_n^k for the determinantal averaging

$$a_n^k = \det(\hat{\mathbf{H}}_{S_n^k}(\boldsymbol{\theta}))$$

$$\mathbf{N}_n^k = a_n \hat{\mathbf{H}}_{S_n^k}^{-1}(\boldsymbol{\theta}) \mathbf{g}_n$$

$$\mathbf{D}_n^k = a_n$$

 Aggregate distorted $\hat{\mathbf{N}}_k$ and $\hat{\mathbf{D}}_k$

$$\hat{\mathbf{N}}_k = \frac{1}{N} \sum_{n=1}^N h_n^k \mathbf{N}_n^k + \mathbf{w}_1^k$$

$$\hat{\mathbf{D}}_k = \frac{1}{N} \sum_{n=1}^N h_n^k \mathbf{D}_n^k + \mathbf{w}_2^k$$

 Update $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \beta \frac{\hat{\mathbf{N}}_k}{\hat{\mathbf{D}}_k}$
 $k \leftarrow k + 1$
output: $\boldsymbol{\theta}_k$

Hessian. The NewSamp algorithm determines both the Hessian and its inverse in $\mathcal{O}(b_n d + (|S_n|_{max} + r)d^2)$ operations per node Erdogdu and Montanari [2015].

The Algorithm: DANDA

Our proposed approximate Newton algorithm, DANDA, is based on the determinantal averaging, adapted to the analog transmission over a wireless fading MAC channel, as described in Algorithm 1. It takes a convergence parameter ϵ as input, which is a threshold for convergence based on the fractional difference between two consecutive values of $F(\boldsymbol{\theta}_k)$. The optimal PDF $\alpha^{*,b}$ is the α value that takes minimum total power per node for convergence.

2.4 Experimental Results

In this section, we provide numerical examples to illustrate the performance of the algorithms DNDA (with and without RLS) and DANDA and their comparison with GBMA

^bWe determined α^* by exhaustive search, varying α within a suitable range in $(0, 1)$, although more sophisticated methods can be used.

2.4. EXPERIMENTAL RESULTS

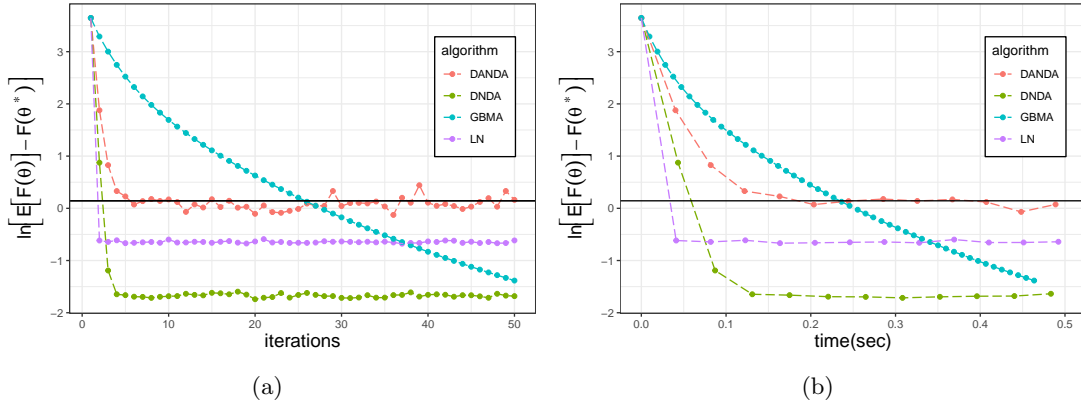


Figure 2.2: Simulation results on (a) convergence and (b) computation time averaged over 100 channel realizations performed on a 1.6 GHz Dual-Core Intel Core i5 processor machine with 8 GB 2133 MHz LPDDR3 RAM

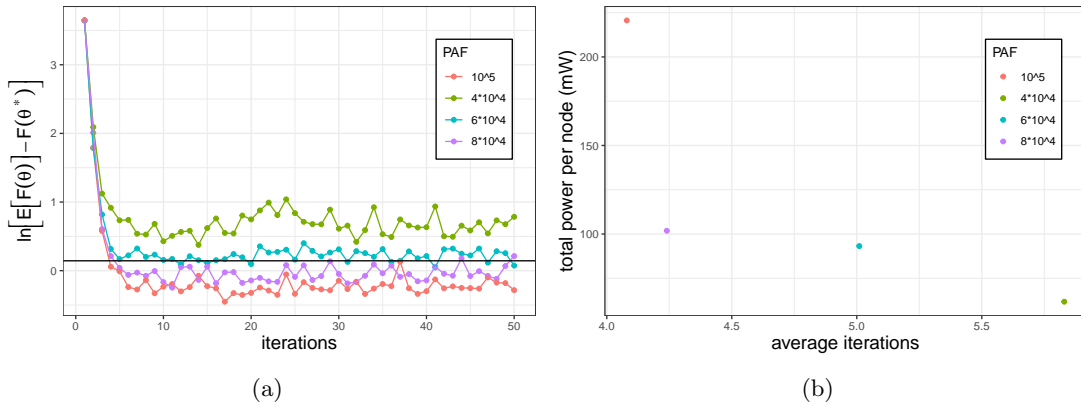


Figure 2.3: Simulation results for (a) Convergence performance of DANDA with different power adjustment factor (PAF) values for DANDA and (b) Power requirements vs convergence rate performance of DANDA averaged over 100 channel realizations

2.4. EXPERIMENTAL RESULTS

Parameter	Value
Network	
AWGN	$w_k \sim \mathcal{N}(\mu = 0, \sigma_w^2 = 4 \text{ mW})$
Channel Fading	$h_n \sim \text{Rayleigh}(\tilde{\sigma}_h = 1)$
Number of nodes	$N = 100$
DANDA	
PAF	7×10^4
Learning Rate	$\beta = 0.95$
Optimal PDF	$\alpha = 0.3$
Sub-Sample size	$ \mathcal{S}_n = 3000$
rank(sub-sampled Hessian)	$r = 50$
DNDA	
PAF	8×10^{24}
Learning Rate	$\beta = 0.95$
Optimal PDF	$\alpha = 0.3$
LN	
PAF	2.5
Learning Rate	$\beta = 1$
L	1
GBMA	
PAF	0.3
Learning Rate	$\beta = 0.1$

Table 2.1: List of Simulation Parameters

(Sery and Cohen [2020]) and LocalNewton (Gupta et al. [2021]) (NOTE: In LN the local parameter estimates are updated L times using Newton’s algorithm before being communicated to the edge server using analog transmission for aggregation). We used the Million Song Dataset (MSD) Bertin-Mahieux et al. [2011] that has 90 audio features for 515,345 soundtracks to recognize the year of song release. We pre-process the data by omitting cases corresponding to the year below 1976 to avoid skewness and outliers in the distribution. The features are normalized to have zero mean and unity variance, and the response has zero mean.^c We split the dataset into training and test sets based on the partition recommendation Dheeru and Taniskidou [2017]. The simulated distributed network has an edge server and 100 nodes, each with a disjoint set of 4,200 data points obtained via uniform random sampling.

We considered a regularized least squares loss function (2.10) with regularization parameter, $\lambda = 0.1$. Table 2.1 shows the value of various simulation parameters for the experiments. The transmitted signals from the N nodes face an i.i.d. Rayleigh fading

^cThis normalization method eliminates the need for the intercept in the linear regression model.

2.4. EXPERIMENTAL RESULTS

algorithm	average time per iteration per node
GBMA	0.009462 s
LN	0.04112 s
DNDA	0.04379 s
DANDA($r = 50, s_n = 3000$)	0.04084 s

Table 2.2: Average computer time (seconds) taken by algorithms per iteration per node on a 1.6 GHz Dual-Core Intel Core i5 processor machine with 8 GB 2133 MHz LPDDR3 RAM

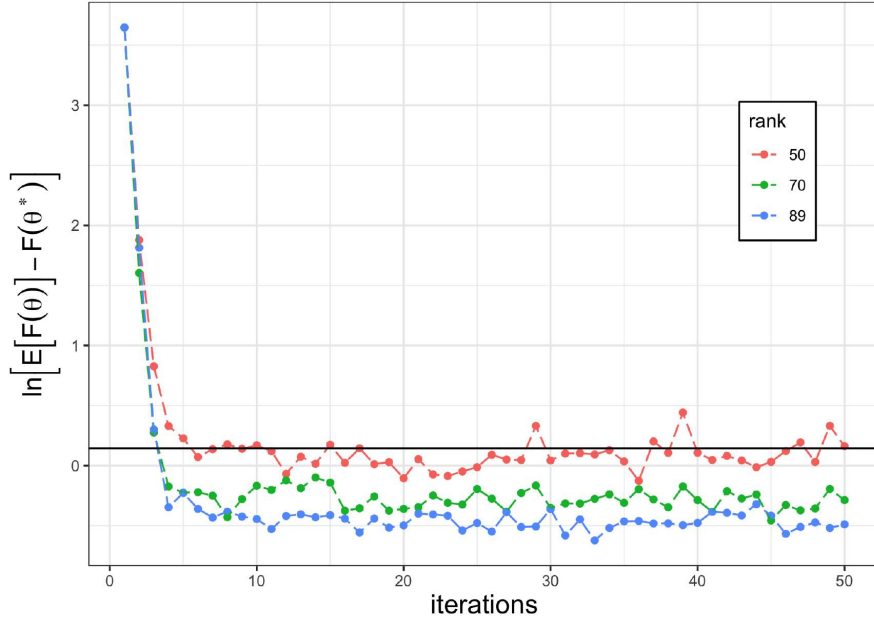


Figure 2.4: Convergence comparison of DANDA with different rank values, keeping every other parameter constant

channel gain with the distribution $p(|h|) = \frac{|h|}{\tilde{\sigma}_h^2} e^{-\frac{|h|}{\tilde{\sigma}_h^2}}$, where $\mu_h = \tilde{\sigma}_h \sqrt{\frac{\pi}{2}}$ and $\sigma_h^2 = \frac{\tilde{\sigma}_h^2(4-\pi)}{2}$.

Fig. 2.2 shows a comparison of DANDA with LN, GBMA, and DNDA. The transmission parameters are set such that each algorithm takes approximately equal total energy for convergence.^d In Fig. 2.2 (a) LN converges the fastest, followed by DNDA and DANDA. The communication channel gain and AWGN affect DANDA the most, while DNDA and GBMA converge to the lowest value, GBMA requiring four times as many iterations to reach convergence. Fig. 2.2 (b) shows the computation time taken to converge by these algorithms. The GBMA takes the least time for computation per iteration as it is gradient-based. The DANDA takes a slightly shorter time for each iteration than LN and DNDA because it uses the approximate Hessian and hence is computationally less

^dThe convergence criteria for the simulation is $\frac{|F(\theta) - F(\theta^*)|}{F(\theta^*)} \leq 0.05$, where $F(\theta^*)$ is the optimal value of centralized newton method. The black horizontal line in the Figures shows this convergence threshold.

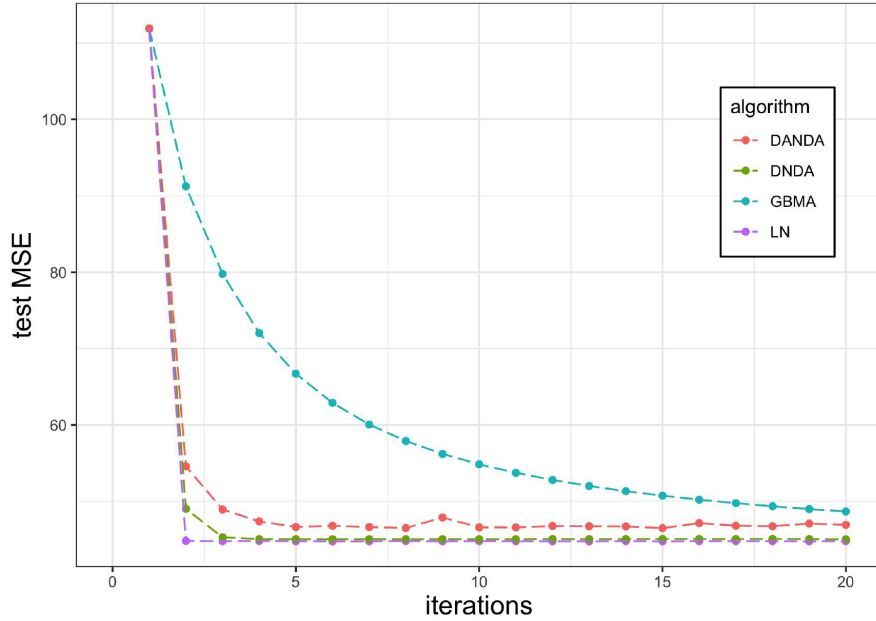


Figure 2.5: Test data performance of DANDA, DNDA, GBMA, and LN

expensive at nodes. Table 2.2 shows the average time per iteration (in seconds) required by the four algorithms. The computations are performed on a 1.6 GHz Dual-Core Intel Core i5 processor machine with 8 GB 2133 MHz LPDDR3 RAM.

Fig. 2.3 shows the performance of the DANDA algorithm with different power settings. In Fig. 2.3 (a), convergence improves with PAF,^e as increasing PAF effectively increases the signal-to-noise ratio (SNR). Fig. 2.3 (b) shows the trade-off between the total power required per node and the average number of iterations the algorithm takes for convergence.

The DANDA algorithm uses Hessian approximation by sub-sampling local data and using the top r eigenvalues from the eigenvalue decomposition, indicated by the rank. Fig. 2.4 shows a comparison of DANDA with different rank values keeping $|S_n| = 3000$. As the rank values increase, the Hessian approximation moves closer to the exact Hessian, and hence the algorithm's performance improves.

Fig. 2.5 shows the accuracy performance of DANDA and other algorithms on the test dataset. All three algorithms are based on second-order optimization and perform similarly on the test data, with LN showing the best results. The GBMA, once again, takes four times as many iterations to achieve its lowest mean squared error (MSE).

The total power consumed for convergence versus the average number of iterations required for DANDA compared to the DNDA, LN, and GBMA is presented in Fig. 2.6.

^eThe actual transmission power is the product of PAF and squared-norm of the vector to be transmitted, eg. in GBMA it is $P\|\mathbf{g}_n(\boldsymbol{\theta}_k)\|_2^2$

2.4. EXPERIMENTAL RESULTS

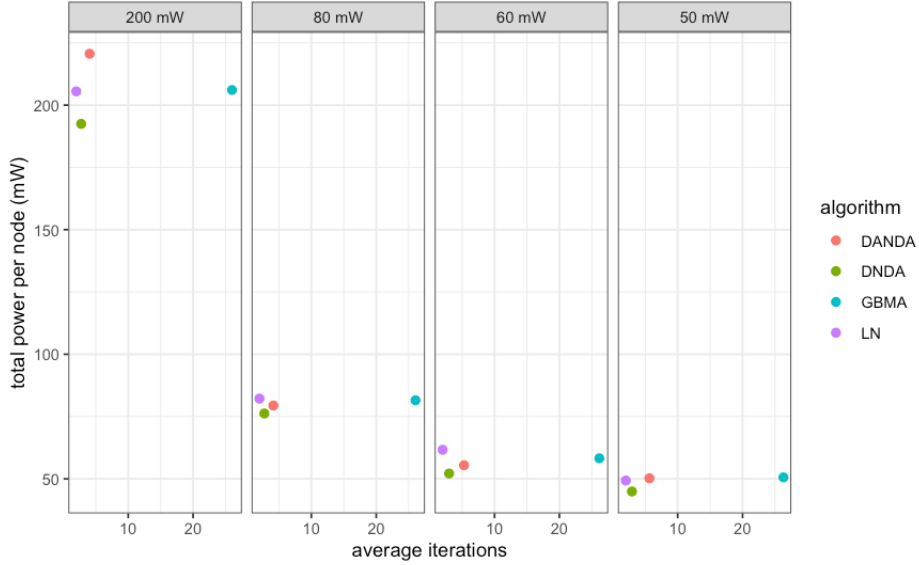


Figure 2.6: Total power vs average iterations comparison of DANDA, DNDA, LN, and GBMA

DANDA, DNDA, and LN take a similar number of iterations for a certain power consumption value,^f while GBMA requires more iterations. Among the second-order methods, LN takes the minimum number of iterations for a given power consumption value, and DANDA takes the maximum. This is because LN transmits a single vector, whereas DNDA (and DANDA) transmits two signals, making them more prone to noise as the ratio of two distorted signals magnifies effective noise.

Fig. 2.7 compares DNDA and DNDA based on RLS and a combination of both. The recursive least squares estimation performs better than DNDA asymptotically, but the convergence is slower. We used a combination of DNDA and RLS-based DNDA, where, after the first iteration, the algorithm switches from DNDA to RLS-based DNDA. This method is useful as the RLS-based DNDA can be initialized by the parameter values obtained after the first iteration of DNDA rather than with an arbitrary value.

NOTE: All experiments presented in this thesis were conducted using simulations on a single computer. The setup does not model real-world edge networks or effects such as latency and device heterogeneity. As a result, the reported results reflect algorithmic performance under idealized conditions. Deployment on physical networks was not attempted, as it would require substantial systems engineering beyond the scope of the present study. Nevertheless, the simulation-based evaluation provides valuable insights into the algorithm's convergence behavior and communication efficiency.

^fThe power consumption values shown in Fig. 6 are approximate and comparable but not identical for all algorithms due to the random channel gain and noise which affects $\mathbf{g}_n(\theta_k)$ and $\mathbf{H}_n(\theta_k)$ at every k^{th} iteration. The results shown in Fig. 6 are averaged over 100 channel realizations

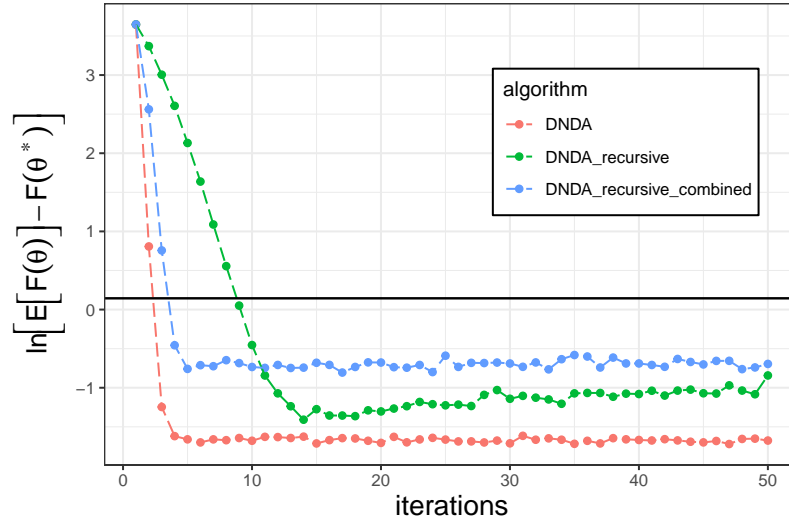


Figure 2.7: Convergence performance of DNDA, DNDA with RLS, and a combination of both

2.5 Conclusions

We propose a distributed machine learning algorithm, DANDA, which combines determinantal averaging with low-rank Hessian approximation at each node and leverages over-the-air edge learning. DANDA achieves faster convergence than the state-of-the-art gradient-based method (GBMA) while reducing the computational cost per iteration through approximation techniques. In addition, it performs competitively with the exact Hessian-based Newton method (LN) in terms of total power consumption and computational time. Finally, DANDA provides stronger privacy guarantees, since the parameter vector θ is never transmitted.

3

Over-the-air Lazy-Hessian based Federated Learning

3.1 Introduction

In recent years, numerous large-scale data-related challenges and opportunities have emerged, necessitating distributed computation to create effective solutions. Federated learning (FL) is one such distributed Machine Learning method where numerous clients work together to jointly train a model while the training data is kept decentralized McMahan et al. [2017a]. Although FL provides a solution to many problems, including data privacy, network congestion, computation cost, etc., there are multiple active areas of improvement in deploying a practical FL algorithm, for example, communication-related challenges like bandwidth and latency, computational constraints at the nodes, and data heterogeneity, etc Chen et al. [2021a].

3.1.1 Related Work

There have been numerous recent research activities in the area of FL focusing on the communication aspect. The early works include gradient-based optimization methods such as FedAvg (McMahan et al. [2017a]), DSGD (Amiri and Gündüz [2019]), and GBMA (Sery and Cohen [2020]). Gradient-based optimization methods are naturally well-suited for distributed computation, but they require a large number of iterations to converge to a reasonable training loss. As a result, second-order optimization methods have gained attention for their ability to achieve faster convergence in fewer iterations, albeit at the expense of higher computational costs from Hessian calculations. To mitigate this overhead, stochastic Newton methods, which approximate the Hessian–vector product (HVP) using sub-sampled Hessians or stochastic curvature estimates, have been extensively studied as an efficient alternative that avoids explicit matrix inversion. For example, LiSSA (Agarwal et al. [2017]) employs a stochastic recursion to estimate $H^{-1}G$, while sub-sampled Newton (Byrd et al. [2012], Roosta-Khorasani and Mahoney [2016]) leverages curvature from random mini-batches. Derezinski and Mahoney [2019] proposed determinantal av-

eraging for distributed inverse-Hessian estimation. Other approaches exploit sketching to compress curvature information Pilanci and Wainwright [2017] and non-uniform sampling schemes to prioritize informative Hessian blocks, offering improved error-complexity trade-offs Xu et al. [2016].

Numerous works in FL setting use quasi-newton and approximate newton methods to reduce the computation cost of Hessian calculation at the nodes, eg. using classical Richardson iteration (DONE) Dinh et al. [2022], using conjugate gradient methods (GIANT and DiSCO)Zhang and Lin [2015], Wang et al. [2018], using Bregman divergence, (DANE) Shamir et al. [2014], etc. One method of reducing the communication overhead is using the Newton method to optimize the parameter vector locally and share the parameter vector updates as done by the authors in (LocalNewton) Gupta et al. [2021]. Authors in Dal Fabbro et al. [2022] and Doikov et al. [2023] used the Lazy-Hessian strategy, where the Hessian is reused to achieve computational efficiency.

Multiple communication strategies have been investigated in the literature including the over-the-air analog communication that has recently been used a lot owing to its natural aggregation upon transmission which suits FL Amiri and Gündüz [2020a,b], Wang et al. [2018], Yang et al. [2020], Zhu et al. [2019], Amiri and Gündüz [2019], Sery and Cohen [2020]. In the presence of channel fading, the transmitted signal experiences distortion, necessitating power regulation. The authors in Zhu et al. [2019] performed power control using “truncated channel inversion,” where a sub-channel is inverted when its gain exceeds a threshold, else disregarding the sub-channel by allocating zero power. In Cao et al. [2019] and Zhang and Tao [2021] authors jointly optimize transmit power at nodes and a signal scaling factor (called denoising factor) subject to individual average power constraints at nodes, in the latter, the authors show that the optimal transmit power at each device is continuous and monotonically decreases with the squared multivariate coefficient of variation of gradient vectors.

3.1.2 Our Contributions

In our research, we extend the Distributed Approximate Newton with Determinantal Averaging (DANDA)Sharma and Dey [2022a] to non-quadratic loss functions with average node power constraint and channel inversion. We explore adaptive node scheduling and transmission power control at the nodes. We call this new algorithm as Distributed Approximate Newton with Channel Inversion and Determinantal Averaging (DANCIDA). Additionally, we examine a “Lazy” variant of the algorithm, where the Hessian is computed only if there is enough variation in the successive gradient vectors. This version is referred to as Lazy-DANCIDA.

This distributed learning algorithm is empirically tested in a logistic regression task

with the Coverttype Dheeru and Taniskidou [2017] dataset, in a simulated communication setting, with its performance averaged over multiple random channel realizations. Comprehensive numerical results illustrate favorable convergence of lazy-DANCIDA compared to state-of-the-art distributed learning methods with comparable energy consumption.

Organization: The rest of the chapter is organized as follows. Section 3.2 talks about the system model and the communication strategy used, and Section 3.3 provides the theoretical background of the algorithm, followed by a description of its implementation. Section 3.4 provides a convergence analysis of the DANCIDA algorithm. Section 3.5 provides a comprehensive set of experimental results on its comparative performance with existing algorithms, followed by some concluding remarks in Section 3.6.

3.2 System Model

The distributed learning model consists of N nodes and an edge server for the aggregation of information coming from the nodes. The aim is to minimize the average of local objective functions as follows.

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^d} \left\{ F(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^N f_n(\boldsymbol{\theta}) \right\} \quad (3.1)$$

where $\boldsymbol{\theta}$ is a $d \times 1$ parameter vector, $f_n(\boldsymbol{\theta})$ is the local objective function of the n^{th} node, and $F(\boldsymbol{\theta})$ is the global objective function, considering equal local sample size at each node.

We consider a similar communication setting as described in Section 2.2.2, where N nodes communicate with an edge server over a wireless MAC channel under Rayleigh block fading. The transmission scheme employs d orthogonal baseband waveforms for analog aggregation. Each node experiences independent channel fading and AWGN at the receiver. For a detailed description of the channel model and signal formulation, refer to Section 2.2.2.

The transmission scheme is divided into two phases as shown in Fig. 3.1. In the first phase, each node calculates a quantity called the local Power Adjustment Factor (PAF) $P_{n,k}$ based on the local dataset for the n^{th} node and the k^{th} iteration, and shares it with the edge server. The edge server then calculates the global PAF $P_k = \min\{P_{1,k}, P_{2,k}, \dots, P_{N,k}\}$ and sends it back to the nodes. This phase guarantees the iterative update of the global PAF to achieve satisfactory signal strength.^a In the second phase, all nodes transmit a

^aWhile Phase 1 holds significance for determinantal averaging, it is not explicitly necessary for trace averaging, as the local trace values exhibit minimal variation across iterations.

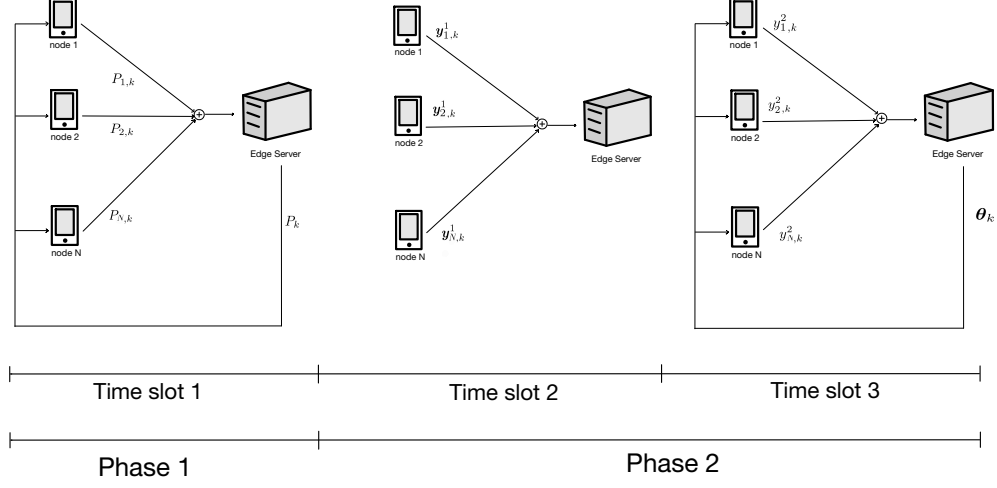


Figure 3.1: Transmission Scheme

linear combination of d-amplified orthogonal analog signals as follows

$$\sqrt{P_k} e^{-j\phi_{n,k}} \mathbf{y}_{n,k} \mathbf{s}(t)$$

where, P_k is the global PAF for the k^{th} iteration received from the edge server after the end of Phase 1 of the transmission and $e^{-j\phi_{n,k}}$ is the phase correction (inverse of estimated channel phase) added to receive positive channel gains at the receiver and \mathbf{y}_n is the information to be transmitted. The node information $\mathbf{y}_{n,k}$ is transmitted using two-time slots sending signals $\mathbf{y}_{n,k}^1$ and $\mathbf{y}_{n,k}^2$ respectively.

According to the block fading model, we assume the fading channel coefficients are constant within a single iteration but change from iteration to iteration in a statistically independent manner.

3.3 The Algorithm

We aim to solve the empirical risk minimization problem described in (3.1) where $f_n(\cdot) : \mathbb{R}^d \mapsto \mathbb{R}$, for all $n \in [N]$. The function $f_n(\cdot)$ satisfies the following assumptions:

Assumption 3.1. $f_n(\cdot)$ is strongly convex and twice differentiable. i.e., there exists

$m > 0$, such that

$$\nabla^2 f_n(\cdot) \succcurlyeq mI$$

Assumption 3.2. $f_n(\cdot)$ has a Lipschitz continuous Hessian: For any $n \in [N]$, there exists a constant L , such that $\forall \boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathbb{R}^d$,

$$\|\mathbf{H}_n(\boldsymbol{\theta}) - \mathbf{H}_n(\boldsymbol{\theta}')\|_2 \leq L\|\boldsymbol{\theta} - \boldsymbol{\theta}'\|$$

where, $\mathbf{H}_n(\boldsymbol{\theta}) = \nabla^2 f_n(\boldsymbol{\theta})$

3.3.1 Newton Direction Estimation

Determinantal Averaging

We adopt the second-order distributed optimization framework introduced in Chapter 2, where the Newton direction is estimated using determinantal averaging Derezhinski and Mahoney [2019]. In particular, the estimator $\hat{\mathbf{p}} = \widehat{\mathbf{H}^{-1}(\boldsymbol{\theta})\mathbf{G}}$ is obtained as a weighted combination of local inverse Hessians, ensuring the absence of inversion bias compared to uniform averaging. A detailed discussion of this estimator is provided in Section 2.3.1.

$$\hat{\mathbf{p}} = \widehat{\mathbf{H}^{-1}(\boldsymbol{\theta})\mathbf{G}}(\boldsymbol{\theta}) = \frac{\sum_{n=1}^N a_n \mathbf{H}_n^{-1}(\boldsymbol{\theta})\mathbf{G}(\boldsymbol{\theta})}{\sum_{n=1}^N a_n} \quad (3.2)$$

Trace-Based Averaging

We introduce another method to estimate the Newton direction using the trace of the local Hessian called Trace-Based Averaging. The estimate of Newton direction in (3.2) is modified as follows:

$$\hat{\mathbf{p}} = \widehat{\mathbf{H}^{-1}(\boldsymbol{\theta})\mathbf{G}}(\boldsymbol{\theta}) = \frac{\sum_{n=1}^N \text{tr}(\mathbf{H}_n) \mathbf{H}_n^{-1}(\boldsymbol{\theta})\mathbf{G}(\boldsymbol{\theta})}{\sum_{n=1}^N \text{tr}(\mathbf{H}_n)} \quad (3.3)$$

Although using the trace of the local Hessian is computationally simpler compared to the determinant, as the latter involves calculating the product of d eigenvalues, the Trace-Based Averaging leads to an asymptotically inconsistent Newton direction estimator Derezhinski and Mahoney [2019].

3.3.2 Node Scheduling and Power Control

The distributed network has N nodes with each node carrying a subset $\mathcal{S}_n \subset \mathcal{R}$ of size b_n (i.e., $b_n = |\mathcal{S}_n|$) of the original dataset, where \mathcal{R} is the original dataset. The local subsets \mathcal{S}_n are disjoint sets i.e., $\mathcal{S}_i \cap \mathcal{S}_j = \emptyset, \forall i, j \in [N], i \neq j$.

We incorporate a node scheduling policy, assuming the Channel State Information (CSI) is known, wherein a node transmits the information only if the individual node's channel gain is higher than the channel gain threshold value ϕ_h and if the channel gain is higher than ϕ_h , we used channel inversion to cancel the effect of channel gain on the transmitted information. The channel scheduling constant $C_{n,k}$ for the n^{th} node and the k^{th} iteration is given as

$$C_{n,k} = \begin{cases} \frac{1}{h_{n,k}} & \text{if } h_{n,k} > \phi_h \\ 0 & \text{otherwise} \end{cases}$$

The channel gain threshold, ϕ_h , is calculated using the average node power constraint as follows.

$$\int_{\phi_h}^{\infty} P_k h_{n,k} dh \leq P_{avg}$$

To ensure sufficient Signal-to-Noise Ratio (SNR), a local power adjustment factor is calculated by the nodes given as follows:

$$P_{n,k} = \frac{P_0}{\|C_{n,k} a_{n,k} \mathbf{H}_{n,k}^{-1}(\boldsymbol{\theta}_k) \mathbf{g}_{n,k}(\boldsymbol{\theta}_k)\|^2 + \|C_{n,k} a_{n,k}\|^2}$$

Where P_0 is the power constant value given to the algorithm to adjust the SNR, $a_{n,k} = \det(\mathbf{H}_{n,k}(\boldsymbol{\theta}_k))$. The local power adjustment factors, $P_{n,k}$, are inversely proportional to the power of the message sent by respective nodes.

The nodes then transmit individual $P_{n,k}$ values to the edge server. The edge server finds the global power adjustment factor $P_k = \min\{P_{1,k}, P_{2,k}, \dots, P_{n,k}\}$ and transmits it back to the nodes.

NOTE: The PAF is communicated by nodes to the edge server and back to nodes via control channels and is not subject to the constraints of the over-the-air aggregation phenomenon.

After receiving the global power adjustment factor from the edge server, the signals to be transmitted are evaluated at the nodes in two separate time slots as follows:

$$\mathbf{y}_{n,k}^1(\boldsymbol{\theta}_k, t) \triangleq \sqrt{\alpha P_k} C_{n,k} e^{-j\phi_{n,k}} a_{n,k} (\mathbf{H}_{n,k}^{-1}(\boldsymbol{\theta}_k) \mathbf{g}_{n,k}(\boldsymbol{\theta}_k))^T \mathbf{s}(t) \quad (3.4)$$

$$t_m < t < t_m + T/2$$

$$\mathbf{y}_{n,k}^2(\boldsymbol{\theta}_k, t) \triangleq \sqrt{(1-\alpha) P_k} C_{n,k} e^{-j\phi_{n,k}} a_{n,k} \mathbf{s}'(t) \quad (3.5)$$

$$t_m + T/2 < t < t_m + T$$

The global PAF P_k value is shared among the two signals ($\mathbf{y}_{n,k}^1$ and $\mathbf{y}_{n,k}^2$) using $\alpha \in (0, 1)$. The $\mathbf{s}(t)$ is the vector of orthogonal normalized waveforms to carry the numer-

3.3. THE ALGORITHM

ator, and $s'(t)$ is a scalar normalized waveform to carry the denominator part of (3.2) respectively.

Signals received at the edge server are superimposed on each other due to the analog nature of the transmitted signals. The two received signals in the first and second halves of the k -th time slot are, respectively:

$$\mathbf{r}_k^1(t) = \sum_{n=1}^{D_k} \sqrt{\alpha P_k} a_{n,k} (\mathbf{H}_{n,k}^{-1}(\boldsymbol{\theta}_k) \mathbf{g}_{n,k}(\boldsymbol{\theta}_k))^T \mathbf{s}(t) + \mathbf{w}_k^1(t), \quad (3.6)$$

$$r_k^2(t) = \sum_{n=1}^{D_k} \sqrt{(1-\alpha) P_k} a_{n,k} s'(t) + w_k^2(t) \quad (3.7)$$

Here \mathbf{w}_k^1 and w_k^2 are AWGN processes $\mathcal{N}(0, \sigma_w^2 \mathbf{I}_d)$ and $\mathcal{N}(0, \sigma_w^2)$ respectively. D_k is the number of nodes selected with the channel scheduling policy.

After matched filtering and scaling, the demodulated signals at the edge server are:

$$\mathbf{v}_k^1 = \frac{1}{D_k} \sum_{n=1}^{D_k} a_{n,k} \mathbf{H}_{n,k}^{-1}(\boldsymbol{\theta}_k) \mathbf{g}_{n,k}(\boldsymbol{\theta}_k) + \tilde{\mathbf{w}}_k^1 \quad (3.8)$$

$$v_k^2 = \frac{1}{D_k} \sum_{n=1}^{D_k} a_{n,k} + \tilde{w}_k^2 \quad (3.9)$$

where $\tilde{\mathbf{w}}_k^1 \sim \mathcal{N}(0, \frac{\sigma_w^2 \mathbf{I}_d}{D_k^2 \alpha P_k})$ and $\tilde{w}_k^2 \sim \mathcal{N}(0, \frac{\sigma_w^2}{D_k^2 (1-\alpha) P_k})$. The edge server receives a superposition of the signals corresponding to the numerator and the denominator in (3.2) from the nodes, along with channel distortion and noise. The ratio of the two signals, $\frac{\mathbf{v}_k^1}{v_k^2}$, gives noisy and distorted version of Newton direction, $\hat{\mathbf{p}}$, which is used for parameter updates as follows,

$$\begin{aligned} \boldsymbol{\theta}_{k+1} &= \boldsymbol{\theta}_k - \beta \frac{\mathbf{v}_k^1}{v_k^2} \\ &= \boldsymbol{\theta}_k - \beta \frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_{n,k} \mathbf{H}_{n,k}^{-1}(\boldsymbol{\theta}_k) \mathbf{g}_{n,k}(\boldsymbol{\theta}_k) + \tilde{\mathbf{w}}_k^1}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_{n,k} + \tilde{w}_k^2} \end{aligned} \quad (3.10)$$

Where β is an appropriately scaled learning rate.

Note that the communication overhead per node involves a $d+1$ dimensional vector at each iteration, requiring a similar communication overhead compared to gradient-based methods. The difficulty, of course, lies in computing the inverse local Hessian at the nodes, which, in general, costs $O(d^3)$ computations per iteration.

The determinantal/trace averaging step in (3.2)/(3.3) requires nodes to calculate the

3.3. THE ALGORITHM

inverse Hessian on their local dataset. Besides being computationally expensive, finding the inverse of the Hessian can be infeasible, especially in scenarios where the computational capacity of nodes is limited. To address this issue, we adopted the approximation of the inverse Hessian recommended in Erdogdu and Montanari [2015]. This approach effectively reduces the computational burden on the nodes, making it more practical for distributed environments with limited computational resources. We need to include the following assumptions in addition to the assumptions described already.

Assumption 3.3. *Bounded Hessian:* $\forall n = 1, 2, \dots, N$, the Hessian of the function $f_n(\theta)$, i.e., $\nabla^2 f_n(\theta)$, is bounded by absolute constants M and m , i.e., $\max_{n \leq N} \|\nabla^2 f_n(\theta)\| \leq M$ and $m \leq \min_{n \leq N} \|\nabla^2 f_n(\theta)\|$.

Assumption 3.4. *Lipschitz continuous sub-sampled Hessian:* For any subset, $S_n \subset [\mathcal{S}_n]$ in $n \in [N]$, there exists constant L_n , such that $\forall \theta, \theta' \in \mathbb{R}^d$,

$$\|\mathbf{H}_{S_n}(\theta) - \mathbf{H}_{S_n}(\theta')\|_2 \leq L_n \|\theta - \theta'\|$$

where, $\mathbf{H}_{S_n}(\theta) = \nabla^2 f_n(\theta)$ over the subset S_n .

This approach is based on a low-rank approximation of the sub-sampled Hessian (Hessian obtained over the subset of data points of size S_n). The low-rank approximation of local sub-sampled Hessian $\mathbf{H}_{S_n}(\theta)$ is performed by using a truncated singular value decomposition (SVD) along the direction of top r eigenvalues only, as, $\hat{\mathbf{H}}_{S_n^k}(\theta) = \mathbf{U}_r^k(\Lambda_r^k)\mathbf{U}_r^{kT}$, where Λ_r is $r \times r$ diagonal matrix with r largest eigenvalues of the local sub-sampled Hessian $\mathbf{H}_{S_n^k}(\theta)$ and \mathbf{U}_r is a $d \times r$ matrix whose columns are the corresponding eigenvectors of $\mathbf{H}_{S_n^k}(\theta)$. The superscript k represents the iteration number.

For stability, the eigenvalues smaller than r^{th} largest eigenvalue are replaced by $(r+1)^{\text{th}}$ largest eigenvalue of $\mathbf{H}_{S_n^k}(\theta)$,

$$\hat{\mathbf{H}}_{S_n^k}(\theta) = \left\{ \frac{1}{|S_n|} \sum_{i \in S_n} \mathbf{x}_i \mathbf{x}_i^T \ell''(\theta_k^T \mathbf{x}_i) + \lambda \mathbf{I}_d \right\} + \mathbf{U}_{\setminus r}^k (\lambda_{r+1}^k \mathbf{I}_{\setminus r} - \Lambda_{\setminus r}^k) \mathbf{U}_{\setminus r}^k \quad (3.11)$$

Here, $\Lambda_{\setminus r}^k$ and $\mathbf{U}_{\setminus r}^k$ are the remaining eigenvalues and the corresponding eigenvectors outside the top r eigenvalue/eigenvector pairs. The inverse of the above approximate Hessian is then given by

$$\hat{\mathbf{H}}_{S_n^k}^{-1}(\theta) = \lambda_{r+1}^{k-1} \mathbf{I}_d + \mathbf{U}_r^k (\Lambda_r^{k-1} - \lambda_{r+1}^{k-1} \mathbf{I}_r) \mathbf{U}_r^{kT} \quad (3.12)$$

Algorithm 2 DANCIDA/DANCITA

input: $f_n(\cdot)$, $\theta_0 \in \mathbb{R}^d$, λ , β , P_0 , α , S_n^k , ϵ , ϕ_h , γ_r
define: $[U_r, \Lambda_r] = \text{TruncatedSVD}_r(\mathbf{H}_{S_n}(\theta))$ as rank- r truncated SVD of local sub-sampled Hessian $(\mathbf{H}_{S_n}(\theta))$ with $(\Lambda_r)_{ii} = \lambda_i$, $k = 1$, $\theta_k = \theta_0$, $F(\theta_k) = \frac{1}{N} \sum_{i=1}^N f_n(\theta_k)$
while $\frac{|F(\theta_k) - F(\theta_{k+1})|}{F(\theta_k)} \geq \epsilon$ **do**

Phase-1:

Perform node selection based on ϕ_h
Calculate the global PAF(P_k)

Phase-2:

for $n = 1$ **to** N **do**

Compute $\mathbf{g}_{n,k}(\theta) = \nabla f_n(\theta_k)$

Compute $\hat{\mathbf{H}}_{S_n^k}^{-1}(\theta)$
 $a_{n,k} = \det(\hat{\mathbf{H}}_{S_n^k}(\theta))$ ($a_{n,k} = \text{tr}(\hat{\mathbf{H}}_{S_n^k}(\theta))$ for DANTA)

 $\mathbf{y}_{n,k}^1 = a_{n,k} \hat{\mathbf{H}}_{S_n^k}^{-1}(\theta) \mathbf{g}_{n,k}(\theta)$
 $\mathbf{y}_{n,k}^2 = a_{n,k}$

Aggregate distorted \hat{N}_k and \hat{D}_k
 $\hat{\mathbf{r}}_k^1 = \frac{1}{D_k} \sum_{n=1}^{D_k} \mathbf{y}_{n,k}^1 + \mathbf{w}_1^k$
 $\hat{\mathbf{r}}_k^2 = \frac{1}{D_k} \sum_{n=1}^{D_k} \mathbf{y}_{n,k}^2 + \mathbf{w}_2^k$
 $\hat{\mathbf{p}} = \frac{\hat{\mathbf{r}}_k^1}{\hat{\mathbf{r}}_k^2}$

Update $\theta_{k+1} = \theta_k - \beta \hat{\mathbf{p}}$
 $k \leftarrow k + 1$
output: θ_{k+1}

3.3.3 Lazy-Approximate Hessian

We further implemented the Lazy-Hessian approach on the sub-sampled Hessian by updating the sub-sampled Hessian only when the norm of the gradient difference of the last two iterations is sufficiently high, i.e., higher than a predefined threshold, ϕ_{hess} .

$$\frac{\|\mathbf{g}_{n,k} - \mathbf{g}_{n,k-1}\|}{\mathbf{g}_{n,k-1}} > \phi_{hess} \quad (3.13)$$

The Lazy-Hessian approach discourages the need to calculate the Hessian when it doesn't change significantly and uses the previously calculated Hessian in case the criterion is not met.

The computational cost in classical second-order methods is $\mathcal{O}(b_n d^2)$ operations per node for the local Hessian and $\mathcal{O}(d^3)$ operations per node for the inverse of the local Hessian. The NewSamp algorithm determines both the Hessian and its inverse in $\mathcal{O}(b_n d + (|S_n|_{max} + r)d^2)$ operations per node Erdogdu and Montanari [2015]. Adding the Lazy-Hessian approach further reduces the computation cost for the overall algorithm.

Algorithm 3 Lazy-DANCIDA/Lazy-DANCITA

input: $f_n(\cdot)$, $\theta_0 \in \mathbb{R}^d$, λ , β , P_0 , α , S_n^k , ϵ , ϕ_h , γ_r , ϕ_{hess}
define: $[U_r, \Lambda_r] = \text{TruncatedSVD}_r(\mathbf{H}_{S_n}(\theta))$ as rank- r truncated SVD of local sub-sampled Hessian $(\mathbf{H}_{S_n}(\theta))$ with $(\Lambda_r)_{ii} = \lambda_i$, $k = 1$, $\theta_k = \theta_0$, $F(\theta_k) = \frac{1}{N} \sum_{i=1}^N f_n(\theta_k)$
while $\frac{|F(\theta_k) - F(\theta_{k+1})|}{F(\theta_k)} \geq \epsilon$ **do**

Phase-1:

Perform node selection based on ϕ_h
Calculate the global PAF(P_k)

Phase-2:

for $n = 1$ **to** N **do**

Compute $\mathbf{g}_{n,k}(\theta) = \nabla f_n(\theta_k)$
if $\frac{\mathbf{g}_{n,(k-1)}(\theta) - \mathbf{g}_{n,k}(\theta)}{\mathbf{g}_{n,(k-1)}(\theta)} \geq \phi_{hess}$ **then**

┌ compute $\hat{\mathbf{H}}_{S_n^k}^{-1}(\theta)$
else

┌ $\hat{\mathbf{H}}_{S_n^k}^{-1}(\theta) = \hat{\mathbf{H}}_{S_n^{k-1}}^{-1}(\theta)$
 $a_{n,k} = \det(\hat{\mathbf{H}}_{S_n^k}(\theta))$ ($a_{n,k} = \text{tr}(\hat{\mathbf{H}}_{S_n^k}(\theta))$ for Lazy-DANTA)

 $\mathbf{y}_{n,k}^1 = a_{n,k} \hat{\mathbf{H}}_{S_n^k}^{-1}(\theta) \mathbf{g}_{n,k}(\theta)$
 $\mathbf{y}_{n,k}^2 = a_{n,k}$

Aggregate distorted $\hat{\mathbf{N}}_k$ and $\hat{\mathbf{D}}_k$

$$\hat{\mathbf{r}}_k^1 = \frac{1}{D_k} \sum_{n=1}^{D_k} \mathbf{y}_{n,k}^1 + \mathbf{w}_1^k$$

$$\hat{\mathbf{r}}_k^2 = \frac{1}{D_k} \sum_{n=1}^{D_k} \mathbf{y}_{n,k}^2 + \mathbf{w}_2^k$$

$$\hat{\mathbf{p}} = \frac{\hat{\mathbf{r}}_k^1}{\hat{\mathbf{r}}_k^2}$$

Update $\theta_{k+1} = \theta_k - \beta \hat{\mathbf{p}}$
 $k \leftarrow k + 1$
output: θ_{k+1}

3.3.4 The Algorithm: Lazy-DANCIDA/Lazy-DANCITA

The algorithms Lazy-DANCIDA and Lazy-DANCITA are an extension of the DANDA algorithm Sharma and Dey [2022a] to a non-quadratic objective function with an adaptive Hessian update strategy. The algorithm also incorporates a channel inversion strategy based on the channel gain threshold value, ϕ_h , that makes the algorithm more energy efficient. Lazy-DANCIDA/Lazy-DANCITA brings improvements to the computational and convergence properties of DANDA. The algorithm works in two phases, in the first phase, the algorithm selects participating nodes having channel gain more than the threshold ϕ_h , and then each selected nodes calculate the local power adjustment factor, $P_{n,k}$ and the edge server finds the global power adjustment factor P_k for the k^{th} iteration. In the second phase, the algorithm performs a local training process to eventually get an estimated global Newton direction.

3.4 Convergence Analysis

In this section, we establish convergence criteria for the DANCIDA algorithm. The convergence analysis requires the assumption of strong convexity and twice differentiability as mentioned in assumption 3.1 and bounded Hessian in assumption 3.3 on the global Hessian, $H(\boldsymbol{\theta})$, i.e., $mI \preceq \nabla^2 F(\boldsymbol{\theta}) \preceq MI$.

Theorem 3.4.1 (Determinantal Averaging Derezinski and Mahoney [2019]). *Let $\hat{\mathbf{H}} = \frac{1}{b} \sum_{i=1}^N b_i \mathbf{Z}_i + \mathbf{B}$ and $\mathbf{H} = \mathbb{E}[\hat{\mathbf{H}}]$, where \mathbf{B} is a positive definite $d \times d$ matrix and b_i are i.i.d. Bernoulli($\frac{b}{R}$). Moreover, assume that all \mathbf{Z}_i are positive semi-definite, $d \times d$, and rank-1. If $b \geq C \frac{\mu d^2}{\eta^2} \log^3 \frac{d}{\delta}$ for $\eta \in (0, 1)$ and $\mu = \max_i \frac{1}{d} \|\mathbf{Z}_i \mathbf{H}^{-1}\|$, then*

$$\left(1 - \frac{\eta}{\sqrt{N}}\right) \mathbf{H}^{-1} \preceq \frac{\sum_{n=1}^N a_n \mathbf{H}_n^{-1}}{\sum_{n=1}^N a_n} \preceq \left(1 + \frac{\eta}{\sqrt{N}}\right) \mathbf{H}^{-1}$$

with probability at least $1 - \delta$, where $\hat{\mathbf{H}}_1, \hat{\mathbf{H}}_2, \dots, \hat{\mathbf{H}}_N \stackrel{i.i.d.}{\sim} \hat{\mathbf{H}}$, $a_n = \det(\hat{\mathbf{H}}_n)$, C is an absolute constant and b is the expected local sample size.

Lemma 1 (NewSamp (Ye et al. [2021])). *Considering assumptions 3.1 and 3.3 hold, and let $0 < \delta < 1$ and target rank r be given. Let λ_{r+1} be the $(r+1)^{\text{th}}$ eigenvalue of $\nabla^2 F(\boldsymbol{\theta})$. Set the sample size $|S_n| \geq \frac{18M \log(2d/\delta)}{\lambda_{r+1}}$. Then, with probability at least $1 - \delta$, the approximate Hessian $\hat{\mathbf{H}}_{s_n^k}$ as shown in equation 3.11 satisfies*

$$(1 - \epsilon_0) \hat{\mathbf{H}}_{s_n^k} \preceq \mathbf{H}_n(\boldsymbol{\theta}) \preceq (1 + \epsilon_0) \hat{\mathbf{H}}_{s_n^k}$$

where $\epsilon_0 = \max\left(\frac{5\lambda_{r+1} + m}{5\lambda_{r+1} + 3m}, \frac{1}{2}\right)$, M is such that $\max_{1 \leq i \leq N} \|\nabla^2 f_i(\boldsymbol{\theta})\| \leq M$, m is such that $\lambda_{\min}(\nabla^2 F(\boldsymbol{\theta})) \geq m > 0$ and $\mathbf{H}_n(\boldsymbol{\theta})$ is the true local Hessian for n^{th} node.

The inequalities mentioned in the above Theorem 3.4.1 and Lemma 1 are high probability results. Hence, we assume that the sample sizes of the local Hessian calculation and Determinantal Averaging in the DANCIDA algorithm are sufficiently large to make the inequalities almost surely true. Combining Theorem 3.4.1 and the Lemma 1, we propose the following convergence criteria for the DANCIDA algorithm.

Remark 3.1. *Suppose D_k is the number of nodes (out of total N nodes) participating in the communication round for iteration k . Considering the probability p of a channel gain being greater than the threshold, ϕ_h as fixed, the number of nodes selected per iteration can be modeled as a (N, p) binomial random variable. Thus, the expected value of D_k is Np and increases with N . Using the law of large numbers, for some arbitrarily small and fixed c , $0 < c < p$, the probability $P\{D_k > N(p - c)\} \rightarrow 1$ as $N \rightarrow \infty$ Feller [1968]. Thus, it can be deduced that there is a large enough number $N' < N$ such, $D_k > N'$ for all k with probability 1 as $N \rightarrow \infty$.*

Theorem 3.4.2 (Convergence of DANCIDA). *Let the assumptions 3.1 to 3.4 hold and let the local sub-sample size, $|S_n| > \max(\frac{18M \log(2d/\delta)}{\lambda_{r+1}}, C \frac{\mu d^2}{\eta^2} \log^3 \frac{d}{\delta})$. If the learning rate β satisfy the following condition,*

$$\beta < 2 \frac{m}{M} \left(\frac{(1 + \epsilon_0)^2 (1 - \frac{\eta}{\sqrt{N'}})}{(1 - \epsilon_0) (1 + \frac{\eta}{\sqrt{N'}})^2} \right)$$

Then, the following inequality holds with probability at least $1 - \delta$ as $N \rightarrow \infty$,

$$F(\boldsymbol{\theta}_{k+1}) < F(\boldsymbol{\theta}_k), \forall k \tag{3.14}$$

μ, M, δ, η and ϵ_0 are defined as in Theorem 3.4.1 and Lemma 1. ϵ is the stopping criterion for the algorithm, hm i.e., the algorithm converges when $\|\nabla F(\theta)\| \leq \epsilon$

The proof of Theorem 3.4.2 is provided in Appendix A (6). The above result implies convergence of the algorithm since $F(\boldsymbol{\theta}_k)$ is lower bounded away from zero. Note that the convergence analysis of DANCITA, Lazy-DANCIDA, and Lazy-DANCITA are significantly more complex and left for future work.

3.5 Experimental Results

In this section, we evaluate the performance of our algorithm Lazy-DANCIDA and Lazy-DANCITA in comparison with other distributed learning algorithms like LocalNewton (Gupta et al. [2021]),^b GIANT (Wang et al. [2018]) and DANDA (Sharma and Dey [2022a]). We looked at noisy versions of GIANT and DNDA algorithms that we call noisy-GIANT and noisy-GI,-DNDA, respectively, where the global gradient is estimated using distorted local gradients from the nodes under channel effect. All the distributed algorithms are simulated using a constant learning rate instead of using a backtracking line search to avoid extra rounds of communication.

The distributed network comprises 100 nodes and an edge server. The channel gain and phase are modeled using Rayleigh fading with $\tilde{\sigma}_h = 1$ and channel noise variance, $\sigma^2 = 2$.

$$p(|h|) = \frac{|h|}{\tilde{\sigma}_h^2} e^{-\frac{|h|^2}{2\tilde{\sigma}_h^2}}$$

where $\tilde{\sigma}_h^2 = 2\sigma_h^2/(4 - \pi)$.

The performance of algorithms is evaluated on the Coverttype dataset Dheeru and Taniskidou [2017], which is about the type of forest cover with 580K data samples and dimension, $d = 55$. We performed binary classification with 2 classes(cover type = 3 and

^bIn LocalNewton the local parameter estimates are updated $L = 1$ times using Newton's algorithm before being communicated to the edge server using analog transmission for aggregation

3.5. EXPERIMENTAL RESULTS

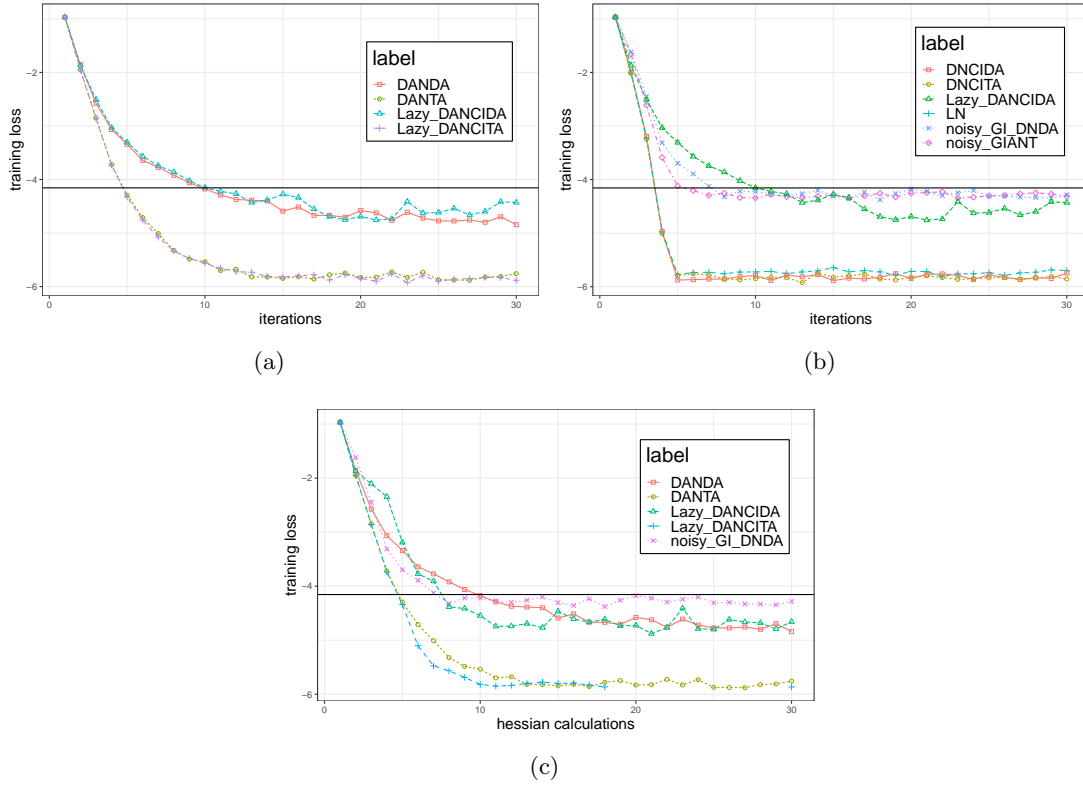


Figure 3.2: Simulation results on convergence and computation time averaged over 100 channel realizations

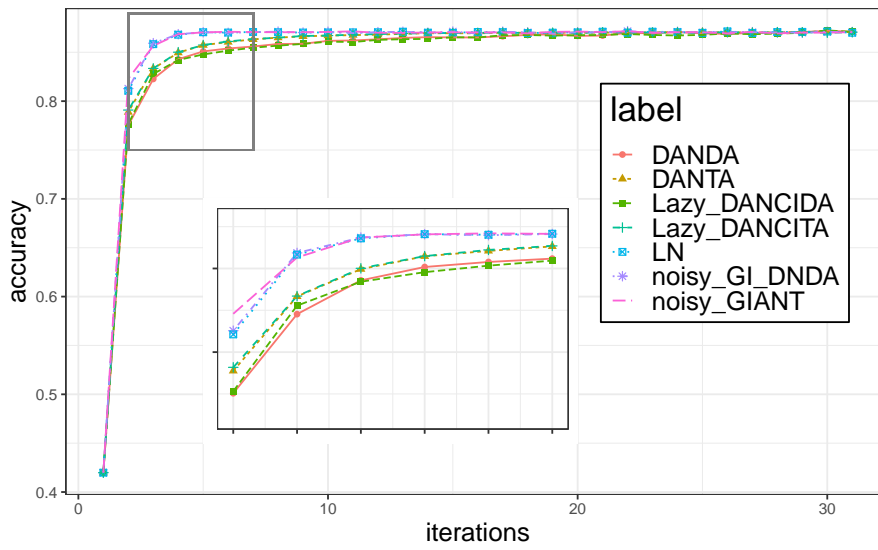


Figure 3.3: Simulation results showing test accuracy of distributed algorithms

cover type = 7) using a regularized logistic regression loss function with regularization parameter, $\lambda = 10^{-4}$. The computations are performed using R statistical software on a 1.6 GHz Dual-Core Intel Core i5 processor machine with 8 GB 2133 MHz LPDDR3 RAM.

In Fig. 3.2, we show a performance comparison of Lazy-DANCIDA and Lazy-DANCITA with multiple distributed learning algorithms. In all figures, the solid horizontal black line shows convergence tolerance,^c, and all the algorithms are constrained to consume almost identical amounts of energy to reach the convergence. Fig. 3.2(a) presents a comparison of convergence among the distributed algorithms utilizing an approximate Hessian for Newton direction calculation. It is intriguing to observe that Lazy-DANCITA and DANCITA take fewer iterations to converge compared to Lazy-DANCIDA and DANCIDA. This shows that the Determinantal averaging is more sensitive to the channel effect than the trace-based averaging for similar power consumption values, as the local Hessian determinant value varies more highly among the nodes than the local Hessian trace value. Fig. 3.2(b) compares Lazy-DANCIDA and Lazy-DANCITA with the distributed algorithms that use the exact Hessian for Newton direction calculations. Despite employing an approximate Hessian, Lazy-DANCIDA, and Lazy-DANCITA perform better than noisy-GI-DNCIDA, but LN, DNCITA, and DNCIDA show superior convergence. In Fig. 3.2(c), Lazy-DANCIDA demonstrates a computational advantage over DANCIDA and noisy-GI-DNCIDA. It requires fewer Hessian calculations to achieve convergence.

Fig. 3.3 shows the test accuracy performance of the distributed learning algorithms with iterations. All the algorithms achieve similar accuracy levels eventually. Notably, LN, noisy-GIANT, and noisy-GI-DNCIDA reach higher accuracy levels a few iterations before DANCIDA, Lazy-DANCIDA, and DANCITA.

The diagram depicted in Fig. 3.4 illustrates the power consumption associated with the distributed learning algorithms for the Hessian calculations done for achieving convergence. It is worth highlighting that LN exhibits the least power consumption vs Hessian calculations to reach convergence because it uses the exact local Hessian for the Newton direction calculation. Lazy-DANCITA and DANCITA display intermediate power usage and Hessian calculation required, and notably least among algorithms using approximate Hessian for the Newton direction calculation. In contrast, noisy-GI-DNCIDA and DANCIDA stand out for both the highest power consumption up to convergence and the largest number of Hessian calculations. The noisy-GI-DNCIDA algorithm, despite using an exact Hessian, consumes high power because of the noisy global gradient.

^cThe convergence tolerance is set to be 5 percent

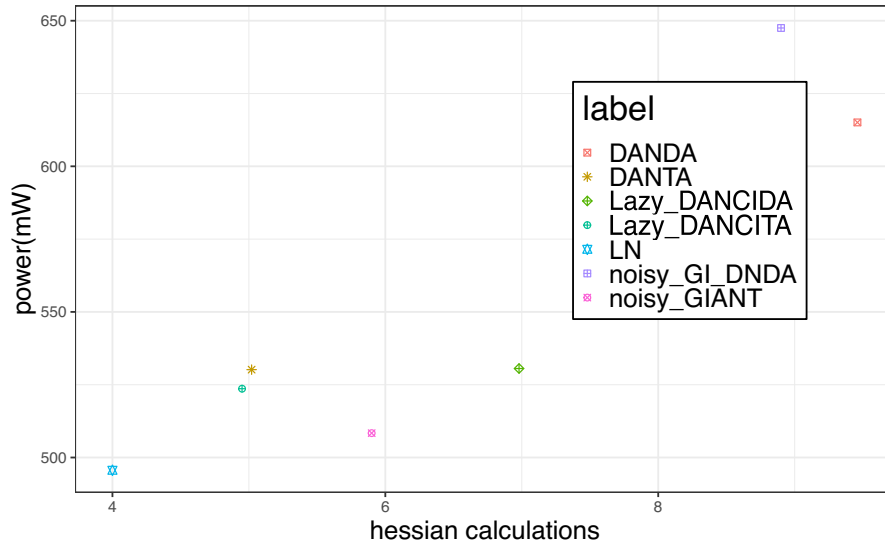


Figure 3.4: Simulation results showing Power required up to convergence for the distributed algorithms

3.6 Conclusions

In this work, we introduced efficient distributed learning algorithms called Lazy-DANCIDA and Lazy-DANCITA. The algorithms achieve communication and computation efficiency by utilizing adaptive Hessian updates and approximate Hessian evaluations, which reduce unnecessary calculations. We adopt an analog over-the-air transmission communication strategy to optimize bandwidth usage. Additionally, we incorporate power control with the channel inversion technique, based on the channel gain threshold, to enhance the algorithm's energy efficiency. Empirical results demonstrate that the algorithm performs well in comparison with other cutting-edge distributed learning algorithms.

4

Fully distributed Newton-type optimization

4.1 Introduction

With billions of IoT devices generating vast amounts of data, often high-dimensional, processing such large volumes on a single machine is challenging. As a result, distributed learning and optimization algorithms have gained significant attention for machine learning tasks. Distributed Machine Learning (DML) enables training across multiple machines using local datasets, where each machine works collaboratively to train a global model without sharing raw data. There are two main frameworks for DML: *(i)* Federated Optimization (also called Federated Learning (FL)), which involves a parameter server acting as a coordinator in a master/worker setting McMahan et al. [2017b], and *(ii)* Networked Optimization (also called fully distributed learning), which has no parameter server and nodes communicate with their single-hop neighbors in a peer-to-peer setting to reach a consensus over shared information. This chapter focuses on the second framework, fully distributed networked learning, specifically emphasizing second-order methods for convex optimization. While second-order methods generally involve higher computational costs per iteration than the first-order methods, they converge in significantly fewer iterations. Consequently, they require fewer communication rounds, often resulting in lower overall communication and computation costs.

4.1.1 Related Work

There has been extensive work in DML in the FL framework, both with gradient-based optimization (see Li et al. [2020d] for a survey) and Hessian-based optimization exploiting approximate Newton-type methods, e.g., DANE (Shamir et al. [2014]), GIANT Wang et al. [2018], DONE (Dinh et al. [2022]), DANDA (Sharma and Dey [2022b]), SHED (Dal Fabbro et al. [2024]), and FedNL (Safaryan et al. [2021]).

Scenarios where a central server is unavailable require resorting to network (also called fully distributed) optimization algorithms, where nodes communicate with each other in

a peer-to-peer fashion to establish consensus. A wide range of communication protocols for achieving distributed consensus can be found in the literature. For example, gossip-based algorithms where nodes communicate with one or few randomly chosen neighbors have been investigated in Kempe et al. [2003], Boyd et al. [2005]. Distributed averaging based on linear iterations, with a suitable weighting matrix, has been implemented in both fixed Xiao and Boyd [2004] and time-varying network topologies Jadbabaie et al. [2003]. Unlike traditional consensus algorithms that only provide asymptotic convergence to the average of the local variables, the finite-time consensus method in Sundaram and Hadjicostis [2007] allows one to compute the exact average in a finite number of steps. Similar results are achieved by the finite-time gossip protocols in Shi et al. [2015b], Chen et al. [2021b]. References Nguyen et al. [2023], Ying et al. [2021] investigate the finite-time consensus properties of some classes of network topology, in some cases by decomposing static graphs into sequences of graphs, and derive conditions under which consensus is reached in $\log_2(N)$ steps, where N is the number of nodes. Finite-time consensus has been investigated in gradient-based distributed optimization, for example, in Charalambous et al. [2015].

The fully distributed learning framework setting has been explored with gradient-based optimization in several works, e.g., Nedic and Ozdaglar [2009], Jakovetić et al. [2014], Yuan et al. [2016], Nedic et al. [2017], whereas the networked version of the Alternating Direction Method of Multipliers (ADMM) has been explored in Shi et al. [2014], Chang et al. [2014]. In the context of Newton-type second-order methods, the Newton-Raphson method has been extended to the network setting in the works Bof et al. [2018a] and Varagnolo et al. [2015]. In particular, the algorithm proposed in the latter either requires the communication of the entire Hessian matrices, which can be prohibitively expensive, or exploits only the diagonal of Hessian matrices, possibly leading to slow convergence for skewed objective functions. The authors in Mokhtari et al. [2016a] and Bajovic et al. [2017] have proposed Network-Newton methods based on an approximate Newton step utilizing a truncated Taylor series expansion of the Hessian inverse but with a penalty-based modified cost function. In Zhang et al. [2021], the authors have proposed a Newton tracking algorithm where each node updates its local variable along a local Newton direction modified with neighboring and historical information. Li et al. [2020e] develop a network version of the DANE algorithm called Network-DANE. The latter requires solving an inner minimization problem each iteration, which can be computationally demanding and can involve hyperparameter tuning for the sub-solver.

4.1.2 Our Contributions

In this chapter, we propose a fully distributed (networked) version of the GIANT (Wang et al. [2018]) algorithm, termed Network-GIANT, which operates without a central server acting as a communication and information aggregator for all the nodes. Instead, the learning takes place over a network (connected graph) of nodes, where each node runs a two-step procedure, leveraging gradient tracking and average consensus. Network-GIANT enjoys a low communication cost per iteration of $K + 1$ d -dimensional vectors for each active edge in the network, where d is the dimension of the problem and $K \geq 1$. We formally prove the semi-global exponential convergence of Network-GIANT to the exact optimal solution.

We further improve the convergence performance of the Network-GIANT algorithm by incorporating an exact consensus algorithm. We term this improved algorithm as Network Exact Consensus-GIANT or NEC-GIANT. The proposed NEC-GIANT takes the best of both worlds: the appealing convergence guarantees of the federated GIANT and the fully distributed property of Network-GIANT.

Finally, we present a set of empirical studies comparing Network-GIANT and NEC-GIANT to other state-of-the-art network learning algorithms, demonstrating their superior convergence performance.

Organization: The remainder of the chapter is organized as follows. Section 4.2 describes the problem and motivation behind our work. The proposed Network-GIANT algorithm and the NEC-GIANT algorithm are presented in Section 4.3, and Section 4.4 details the convergence analysis of the algorithms. Section 4.5 presents numerical simulations on two real datasets, followed by concluding remarks in Section 4.6.

4.2 System Model

Consider a set of agents/nodes $\mathcal{N} = \{1, \dots, N\}$ where each node i has access to a possibly private local cost function $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$. We are interested in solving, in a distributed fashion, the unconstrained optimization problem

$$F(\boldsymbol{\theta}^*) = \min_{\boldsymbol{\theta} \in \mathbb{R}^d} \left\{ F(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^N f_i(\boldsymbol{\theta}) \right\} \quad (4.1)$$

where each $f_i(\boldsymbol{\theta})$ is strongly convex. This general formulation suits several problems that frequently arise in machine learning, in particular, the task of empirical risk minimization. In the latter, $\boldsymbol{\theta} \in \mathbb{R}^d$ parametrizes a model to be learned, and each $f_i(\boldsymbol{\theta})$ is a suitable loss function evaluated over the data samples possessed by agent i . In the context described above, we want to devise a second-order distributed optimization algorithm that exploits

the convexity of the objective functions to achieve faster convergence. Our goal is to reach the exact solution, differently from Bajovic et al. [2017] and Mokhtari et al. [2016a], which can only guarantee convergence to a neighborhood of the latter.

Notation: For a generic variable $\boldsymbol{\theta}$, we use $\boldsymbol{\theta}_i^k$ to denote the local copy of $\boldsymbol{\theta}$ possessed by node i at iteration k . $\|\cdot\|$ is the Euclidean norm, I is the identity matrix and the superscript T indicates the transpose of the argument. $\mathbf{1}$ and $\mathbf{0}$ are d -dimensional vectors whose components are all equal to 1 and 0, respectively.

GIANT. In the Federated Learning setting, the authors of Wang et al. [2018] addressed a class of empirical risk minimization problems and proposed GIANT, a Newton-type method based on the update

$$\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - (H^k)^{-1} \left(\frac{1}{N} \sum_{i=1}^N \nabla f_i(\boldsymbol{\theta}^k) \right), \quad (4.2)$$

$$H^k = \left(\frac{1}{N} \sum_{i=1}^N [\nabla^2 f_i(\boldsymbol{\theta}^k)^{-1}] \right)^{-1}. \quad (4.3)$$

In comparison, the traditional Newton-Raphson recursion relies on the true Hessian of $F(\boldsymbol{\theta})$, given by the arithmetic mean of the local Hessians. According to Wang et al. [2018], if the information is spread out among the data samples, then the harmonic mean H^k is very close to the arithmetic mean, and therefore (4.2) becomes a good approximation of the Newton update. In practice, a line search procedure is needed to select a suitable step size and prevent algorithmic divergence.

GIANT is designed for the master-worker architecture, where all the agents are connected in a star topology to a central aggregator node. The job of the latter is to gather the local information and compute the averages $\nabla f(\boldsymbol{\theta}^k)$ and η , which are respectively the mean of the local gradients and the mean of the approximate Newton directions $\eta_i = \nabla^2 f_i(\boldsymbol{\theta}^k)^{-1} \nabla f(\boldsymbol{\theta}^k)$ $i = 1, \dots, N$. The pseudocode 4 refers to the version of GIANT in which the step size is chosen using the following line search procedure. First, each worker i evaluates its local function at $\boldsymbol{\theta}_i^k - \alpha_j \eta$ for a fixed set of candidate step-sizes $\{\alpha_j\}$. Then, the master selects a step-size α^* that satisfies the Armijo–Goldstein condition, and the workers use it to update their local variable.

Overall, each iteration requires the transmission for each agent of 4 d -dimensional vectors, a vector whose size is the cardinality of $\{\alpha_j\}$, and a scalar, resulting in competitive communication complexity.

Despite being provably more efficient than several first-order and second-order methods, the applicability of GIANT is limited to the master-worker scenario. To overcome this shortcoming, in the next section, we introduce Network-GIANT and NEC-GIANT,

which allow implementation of the update formula (4.2) in a general network of nodes without the need for a central coordinator.

Algorithm 4 GIANT with line search Wang et al. [2018]

Initialize: Choose $\boldsymbol{\theta}^0 \in \mathbb{R}^d$, set $\boldsymbol{\theta}_i^0 = \boldsymbol{\theta}_0 \forall i = 1, 2, \dots, N$.

Choose the set of candidate step-sizes α_j .

for each iteration $k = 0, 1, 2, \dots$ **do**

$\nabla f(\boldsymbol{\theta}^k) = \frac{1}{N} \sum_{i=1}^N \nabla f_i(\boldsymbol{\theta}^k)$ ▷ at the master

for each worker $i = 1, \dots, N$ **do**

$\eta_i = \nabla^2 f_i(\boldsymbol{\theta}^k)^{-1} \nabla f(\boldsymbol{\theta}^k)$

$\eta = \frac{1}{N} \sum_{i=1}^N \eta_i$ ▷ at the master

for each worker $i = 1, \dots, N$ **do**

$f_{i,\alpha_j} = f_i(\boldsymbol{\theta}_i^k - \alpha_j \eta) \quad \forall \alpha_j \in \{\alpha_j\}$

$\alpha^* = \operatorname{argmin}_{\alpha_j} \frac{1}{N} \sum_{i=1}^N f_{i,\alpha_j}$ ▷ at the master

for each worker $i = 1, \dots, N$ **do**

$\boldsymbol{\theta}_i^{k+1} = \boldsymbol{\theta}_i^k - \alpha^* \eta$

4.3 Fully Distributed GIANT

The GIANT algorithm assumes the presence of a master node that can gather all the local information, compute the mean, and send it back to the workers in one shot. In this section, we show how to remove the dependency from such a central entity using tools such as distributed average consensus and gradient tracking. We first provide a formal definition of the multi-agent setting under consideration, and then we show how to modify GIANT to make it suitable for peer-to-peer distributed optimization.

We consider the communication network represented by the connected and time-invariant graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$, where the vertex set \mathcal{N} is the set of agents. The edges \mathcal{E} are the available bi-directional communication links, and each node can only communicate with its single-hop neighbors. The network is associated with a consensus matrix $P \in \mathbb{R}^{N \times N}$, whose generic entry p_{ij} is positive if edge $(i, j) \in \mathcal{E}$ and zero otherwise. The mixing matrix P is symmetric and doubly stochastic, i.e., $P\mathbf{1} = \mathbf{1}$, $\mathbf{1}^T P = \mathbf{1}^T$, the eigenvalues of P lie in $(-1, 1]$ and the null space of $(I - P)$ is $\operatorname{span}(\mathbf{1})$. It is possible to build a matrix that satisfies these requirements without complete knowledge of the graph topology using the Metropolis weights Xiao et al. [2007].

Average Consensus. In this setup, the centralized averages originally computed by the master can be replaced by two average consensus blocks. Consensus is an iterative process where each step is composed of two stages: a communication phase, in which each agent i transmits its local variable $\boldsymbol{\theta}_i$ to its neighbors, followed by a local update, where $\boldsymbol{\theta}_i$ is

4.3. FULLY DISTRIBUTED GIANT

updated using a weighted average of the information received by node i . The weights are given by the consensus matrix P , and this process continues until all agents converge to the same value, corresponding to the arithmetic mean of the initial values of the local variables.

However, the standard average consensus is not suited to estimate the global cost gradient efficiently. We then resort to the technique of average tracking consensus, introducing the local variable

$$w_i^{k+1} = \sum_{j=1}^N p_{ij} \left(w_j^k + \nabla f_j(\boldsymbol{\theta}_j^k) - \nabla f_j(\boldsymbol{\theta}_j^{k-1}) \right). \quad (4.4)$$

Thanks to the doubly stochasticity of the matrix P , choosing any initialization that satisfies $w_i^0 = \nabla f_i(\boldsymbol{\theta}_i^{-1}) \forall i \in \mathcal{N}$ we get the useful property

$$\frac{1}{N} \sum_{i=1}^N w_i^{k+1} = \frac{1}{N} \sum_{i=1}^N \nabla f_i(\boldsymbol{\theta}_i^k). \quad (4.5)$$

As the consensus process proceeds, the local variables w_i will get closer to each other and will track increasingly better the gradient of the overall cost function $F(\boldsymbol{\theta})$.

Algorithm 5 Network-GIANT

Initialize: Arbitrary $\boldsymbol{\theta}_i^0 \in \mathbb{R}^d$

$$w_i^0 = \nabla f_i(\boldsymbol{\theta}_i^{-1}) = \mathbf{0} \quad \forall i \in \mathcal{N}$$

$\beta > 0$, consensus matrix P .

for each iteration $k = 0, 1, 2, \dots$ **do**

for each agent $i = 1, \dots, N$ **do**

$$v_i^{k+1} = w_i^k + \nabla f_i(\boldsymbol{\theta}_i^k) - \nabla f_i(\boldsymbol{\theta}_i^{k-1})$$

$$w_i^{k+1} = \sum_{j=1}^N p_{ij} v_j^{k+1}$$

$$u_i^{k+1} = \boldsymbol{\theta}_i^k - \beta \nabla^2 f_i(\boldsymbol{\theta}_i^k)^{-1} w_i^{k+1}$$

$$\boldsymbol{\theta}_i^{k+1} = \sum_{j=1}^N p_{ij} u_j^{k+1}$$

▷ optional: repeat K times

4.3.1 Network-GIANT

We are now ready to state our distributed second-order algorithm for general networks, which we refer to as Network-GIANT. The local variables of the agents are initialized with arbitrary values of $\boldsymbol{\theta}_i^0 \in \mathbb{R}^d$ and $w_i^0 = \nabla f_i(\boldsymbol{\theta}_i^{-1}) \forall i \in \mathcal{N}$. At each iteration k , each agent i computes the gradient $\nabla f_i(\boldsymbol{\theta}_i^k)$ of its local function and updates w_i^k using (4.4), which involves a consensus step. The resulting w_i^{k+1} is combined with the local Hessian

to obtain an approximate Newton direction and descend along it, obtaining

$$u_i^{k+1} = \theta_i^k - \beta \nabla^2 f_i(\theta_i^k)^{-1} w_i^{k+1}. \quad (4.6)$$

The iteration terminates with a consensus step on the direction u_i^{k+1} , whose result provides the value of θ_i^{k+1} . Compared with GIANT, which employs a line search procedure, in (4.6), we use a fixed step size $\beta > 0$. This implies that at each iteration, only two (or $K + 1$ if the consensus step on u_i is repeated K times) d -dimensional vectors per agent are transmitted, resulting in an appealing communication cost.

The computational complexity of Network-GIANT is dominated by the inversion of the Hessian matrix and the subsequent product with the gradient estimator that appears in (4.6). This computation can be efficiently carried out in a Hessian-free manner, avoiding the explicit storage and inversion of the Hessian Nocedal and Wright [1999]. Since the Hessians of the strongly convex cost functions are guaranteed to be positive-definite, we can leverage the conjugate gradient method to solve the linear systems, obtaining the Newton step cheaply. In this way, we do not require explicit knowledge of the Hessian, and we only need Hessian-vector products, which can also be obtained by automatic differentiation Baydin et al. [2018] or finite differencing Shen et al. [2019].

Remark 4.1. *Aligning with similar literature, e.g., ESOM-K Mokhtari et al. [2016b], Network-DANE for different values of K Li et al. [2020e], NN-K Mokhtari et al. [2016a], also in Network-GIANT, it is possible to substitute single consensus steps with multi-step consensus blocks. For example, the consensus step on the variables $\{u_1, \dots, u_N\}$ can be replaced with $K \geq 1$ consecutive consensus rounds. This is equivalent to a single consensus step in which the mixing matrix P^K is used. The parameter K adds flexibility to the algorithm, introducing a trade-off between communication cost, wall-clock time, and number of iterations needed for convergence.*

Table 4.1 compares the communication cost per iteration of Network-GIANT with other fully distributed second-order methods available in the literature. While the communication complexities per iteration are similar, the numerical simulations in Section 4.5 show that Network-GIANT requires the least amount of data transmitted overall to converge.

4.3.2 NEC-GIANT

In this section, we present a new algorithm called Network Exact Consensus-GIANT (NEC-GIANT), which extends the federated algorithm GIANT (Wang et al. [2018]) to the fully distributed scenario, keeping the convergence properties and iteration complexity unchanged.

4.3. FULLY DISTRIBUTED GIANT

Algorithm	Communication cost
Jacobi NRC Varagnolo et al. [2015]	$2d$
Newton Tracking Zhang et al. [2021]	d
NetworkDANE Li et al. [2020e]	$2Kd$
ESOM Mokhtari et al. [2016b]	$(K + 1)d$
Network-GIANT	$(K + 1)d$

Table 4.1: Communication cost of different second-order fully distributed algorithms, quantified as the number of scalars transmitted along each active edge of the network (i.e., for each non-zero off-diagonal entry of the mixing matrix P) at each iteration. The cost is expressed as a function of the dimension d of the problem and the number K of consensus steps.

We first point out the weakness of Network-GIANT, which is that it does not use exact averages of local variables but rather uses estimates that may be inaccurate, especially at the beginning of the algorithm. In fact, in Network-GIANT, the global gradient is estimated using average consensus tracking, and the global approximate Newton direction is estimated indirectly by performing one or a few steps of standard consensus on the updated decision variable. These consensus values are only asymptotically convergent and constitute a performance bottleneck.

To overcome this problem, we propose an improved, fully distributed version of GIANT based on the Finite-Time Distributed Consensus (FTDC) protocol in Sundaram and Hadjicostis [2007]. The FTDC protocol provides exact consensus values of local quantities and thus helps preserve the original convergence properties of GIANT in a networked setting.

Finite-Time Distributed Consensus. We now introduce the finite-time distributed consensus (FTDC) procedure in Sundaram and Hadjicostis [2007]. At the end of the protocol, each node knows the exact average, similar to what happens in the federated setting, but without needing a central server.

The requirements that a consensus matrix P must satisfy to attain FTDC are the following: a) P has a simple eigenvalue at 1, and all other eigenvalues have magnitude strictly less than 1, b) The left and right eigenvectors of P corresponding to the eigenvalue 1 are $\frac{1}{N}\mathbf{1}^T$ and $\mathbf{1}$, respectively. The theoretical concept underlying the protocol is the minimum polynomial of the matrix P , defined as the unique monic polynomial of the smallest degree $D + 1 \leq N$ that satisfies

$$P_{D+1} + \alpha_D P_D + \dots + \alpha_1 P_1 + \alpha_0 I = 0 \quad (4.7)$$

4.4. CONVERGENCE ANALYSIS

Assume that we want to calculate the average of a generic local variable s_i in a distributed manner. Let $s = [s_1 \cdots s_N]^T$ be the stack of the local variables so that a single step of the FTDC protocol can be written as $s[t+1] = Ps[t]$. Since $s[D+1] = P^{D+1}s[0]$, taking P^{D+1} from (4.7) and considering a generic step $t \in \mathbb{N}$ we obtain the identity

$$s_i[t+D+1] + \alpha_D s_i[t+D] + \cdots + \alpha_1 s_i[t+1] + \alpha_0 s_i[t] = 0 \quad (4.8)$$

Equation (4.8) shows that the values a local variable takes in the last $D+1$ consensus steps contain all the information needed to compute all its future values, without actually performing further consensus steps. Taking the Z-transform of (4.8) and applying the final value theorem gives a closed-form expression of the average:

$$\begin{aligned} \frac{1}{N} \mathbf{1}^T s[0] &= \lim_{t \rightarrow \infty} s_i[t] \\ &= \lim_{z \rightarrow 1} (z-1) S_i[z] \\ &= \frac{\begin{bmatrix} s_i[D] & s_i[D-1] & \cdots & s_i[0] \end{bmatrix} G}{\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix} G}, \end{aligned} \quad (4.9)$$

where,

$$G = \begin{bmatrix} 1 \\ 1 + \alpha_D \\ 1 + \alpha_{D-1} + \alpha_D \\ \cdot \\ \cdot \\ 1 + \sum_{j=1}^D \alpha_j \end{bmatrix}$$

This results in a simple and intuitive pseudocode, shown in Algorithm 6. Similarly to Network-GIANT, when descending along the approximate Newton direction, we allow for a fixed stepsize $\beta > 0$. The algorithm NEC-GIANT follows the steps as shown in Algorithm 7.

4.4 Convergence Analysis

In this section, we prove the semi-global exponential convergence of Network-GIANT to the exact solution of the optimization problem (4.1). The following assumption formalizes the convexity and regularity properties of the objective functions.

Assumption 4.1 (Strong Convexity and Smoothness). *Let the local costs $f_i(\theta) \forall i \in \mathcal{N}$ be*

4.4. CONVERGENCE ANALYSIS

Algorithm 6 FTDC $(s_i[0], G, P)$ (seen from node i)

Initialize: initial value $s_i[0]$ of node i ,
coefficient vector $G \in \mathbb{R}^{D+1}$,
consensus matrix $P \in \mathbb{R}^{N \times N}$.

for each step $t = 0, \dots, D$ **do**

for each node $i \in \mathcal{N}$ **in parallel do**

$$\left[\begin{array}{l} s_i[t+1] = \sum_{j=1}^N p_{i,j} s_j[t] \end{array} \right.$$

$$FTDC(s_i[0], G, P) = \frac{\begin{bmatrix} s_i[D] & s_i[D-1] & \cdots & s_i[0]G \end{bmatrix}}{\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}}$$

Algorithm 7 NEC-GIANT

Initialize: Arbitrary $\theta_i^0 \in \mathbb{R}^d \forall i \in \mathcal{N}$

$\beta > 0$, consensus matrix P ,

Compute the distinct eigenvalues $\lambda_0, \dots, \lambda_D$ of P ,

Compute the coefficients of the minimal polynomial

$$\prod_{i=1}^D (p - \lambda_i) = p^{D+1} + \alpha_D p^D + \cdots + \alpha_1 p + \alpha_0, \alpha_0 \neq 0,$$

Compute G as in (4.9)

for each iteration $k = 0, 1, 2, \dots$ **do**

for each agent $i \in \mathcal{N}$ **in parallel do**

$$\left[\begin{array}{l} g_i^k = FTDC(\nabla f_i(\theta_i^k), G, P) \\ \eta_i^k = FTDC(\nabla^2 f_i(\theta_i^k)^{-1} g_i^k, G, P) \\ \theta_i^{k+1} = \theta_i^k - \beta \eta_i^k \end{array} \right.$$

two times differentiable, μ -strongly convex, and with L -Lipschitz and continuously differentiable gradients. This assumption, which is standard in the field of convex optimization, tells that there exist positive constants μ, L such that $\mu I \leq \nabla^2 f_i(\theta) \leq LI, \forall \theta \in \mathbb{R}^d$. To simplify the proof of Theorem 4.4.2, we further characterize the objective functions, assuming that the inverse Hessian matrices are continuously differentiable functions.

4.4.1 Network-GIANT

Our proof is based on the principle of time-scales separation for discrete-time systems Maritan and Schenato [2023]. To keep the manuscript self-contained and for a better understanding of the proof of our main result, we provide a statement of such theorem.

Theorem 4.4.1 (Separation of time-scales Maritan and Schenato [2023]). *The origin of the autonomous discrete-time system*

$$\begin{cases} z^k = z^{k-1} + \beta \phi(z^{k-1}, y^{k-1}) \\ y^k = \varphi(y^{k-1}, z^{k-1}) + \beta \Psi(y^{k-1}, z^{k-1}) \end{cases} \quad (4.10)$$

is semi-globally exponentially stable for a suitable choice of the parameter $\beta > 0$, provided

4.4. CONVERGENCE ANALYSIS

that all the following conditions are verified: (a) The functions φ , Ψ and ϕ are locally uniformly Lipschitz. (b) There exists y^* such that $y^*(z) = \varphi(y^*(z), z)$ for all z . (c) At the origin, $\phi(0, y^*(0)) = 0$ and $\Psi(y^*(0), 0) = 0$. (d) There exists a twice differentiable function $V(z)$ and positive constants c_1, c_2, c_3, c_4 such that

$$\begin{aligned} c_1 \|z\|^2 &\leq V(z) \leq c_2 \|z\|^2, \\ \frac{\partial V}{\partial z} \phi(z, y^*(z)) &\leq -c_3 \|z\|^2, \quad \left\| \frac{\partial V}{\partial z} \right\| \leq c_4 \|z\|. \end{aligned} \tag{4.11}$$

The characterization ‘‘semi-globally exponentially stable’’ means that Network-GIANT exhibits a linear convergence rate when initialized within a neighborhood of the optimal solution. A more rigorous definition can be found in Proposition 8.1 in Bof et al. [2018b]. Below, we present our main result, which provides theoretical guarantees for the stability and performance of the proposed algorithm. Since the parameter K does not affect the convergence proof, for simplicity, we set $K = 1$.

Theorem 4.4.2 (Convergence of Network-GIANT). *Under Assumption 4.1, there exists a positive $\bar{\beta}$ such that for any $\beta \leq \bar{\beta}$, Network-GIANT converges semi-globally and exponentially fast to the exact solution of (4.1).*

Proof. The proof follows the same steps as Theorem 2 in Maritan and Schenato [2023] but presents different challenges. Our goal is to rewrite Network-GIANT as a discrete-time system compatible with Theorem 4.4.1 and then show that all the assumptions of the latter are satisfied. To obtain a compact notation, we define the stacks of variables $\in \mathbb{R}^{N \times d}$

$$\begin{aligned} \boldsymbol{\theta}^k &:= [\boldsymbol{\theta}_1^k, \dots, \boldsymbol{\theta}_N^k]^T, \quad \nabla f(\boldsymbol{\theta}^k) := [\nabla f(\boldsymbol{\theta}_1^k), \dots, \nabla f(\boldsymbol{\theta}_N^k)]^T, \\ \nabla^2 f(\boldsymbol{\theta}^k)^{-1} w^{k+1} &:= [\nabla^2 f_1(\boldsymbol{\theta}_1^k)^{-1} w_1^{k+1}, \dots]^T. \end{aligned}$$

Without loss of generality, we translate the objective function by an offset so that the solution $\boldsymbol{\theta}^* = \operatorname{argmin} F(\boldsymbol{\theta}) = \mathbf{0}$ and $f(\boldsymbol{\theta}^*) = \mathbf{0}$. We decompose $\boldsymbol{\theta}$ into the sum of its mean $\bar{\boldsymbol{\theta}} = \frac{1}{N} \mathbf{1}\mathbf{1}^T x$, which has all equal rows, and $\tilde{\boldsymbol{\theta}} = \boldsymbol{\theta} - \bar{\boldsymbol{\theta}}$. Using the double stochasticity of the matrix P , it is easy to see that Network-GIANT is equivalent to the discrete-time dynamical system identified by the slow dynamics

$$\begin{aligned} \bar{\boldsymbol{\theta}}^{k+1} &= \bar{\boldsymbol{\theta}}^k + \beta \left(-\frac{1}{N} \mathbf{1}\mathbf{1}^T \nabla^2 f(\bar{\boldsymbol{\theta}}^k + \tilde{\boldsymbol{\theta}}^k)^{-1} w^{k+1} \right) \\ &= \bar{\boldsymbol{\theta}}^k + \beta \phi(\bar{\boldsymbol{\theta}}^k, \xi^k) \end{aligned} \tag{4.12}$$

and the fast dynamics

$$\begin{aligned}
g^{k+1} &= \varphi_g = \nabla f(\bar{\boldsymbol{\theta}}^k + \tilde{\boldsymbol{\theta}}^k), \\
w^{k+1} &= \varphi_w = P \left(w^k + \nabla f(\bar{\boldsymbol{\theta}}^k + \tilde{\boldsymbol{\theta}}^k) - g^k \right), \\
\tilde{\boldsymbol{\theta}}^{k+1} &= \left(I - \frac{1}{N} \mathbf{1}\mathbf{1}^T \right) P \left(\tilde{\boldsymbol{\theta}}^k - \beta \nabla^2 f(\boldsymbol{\theta}^k)^{-1} w^{k+1} \right) \\
&= \varphi_{\tilde{\boldsymbol{\theta}}} + \beta \Psi(\bar{\boldsymbol{\theta}}^k, \xi^k),
\end{aligned} \tag{4.13}$$

where we used the fact that $(I - \frac{1}{N} \mathbf{1}\mathbf{1}^T) P \bar{\boldsymbol{\theta}} = 0$ and defined

$$\begin{aligned}
\xi &= \{g, w, \tilde{\boldsymbol{\theta}}\}, \quad \varphi_{\tilde{\boldsymbol{\theta}}} = \left(P - \frac{1}{N} \mathbf{1}\mathbf{1}^T \right) \tilde{\boldsymbol{\theta}}^k, \\
\Psi(\bar{\boldsymbol{\theta}}^k, \xi^k) &= \left(\frac{1}{N} \mathbf{1}\mathbf{1}^T - P \right) \nabla^2 f(\bar{\boldsymbol{\theta}}^k + \tilde{\boldsymbol{\theta}}^k)^{-1} w^{k+1}.
\end{aligned} \tag{4.14}$$

Using the definition of w^{k+1} we note that the system is autonomous, and setting $z = \bar{\boldsymbol{\theta}}$, $y = \xi$ and $\varphi = \varphi_g + \varphi_w + \varphi_{\tilde{\boldsymbol{\theta}}}$ we get the same decoupled structure of the system (4.10). Below, we show that all the conditions of Theorem 4.4.1 are verified.

Since w is a linear combination of continuously differentiable and Lipschitz continuous terms, we have that w is itself continuously differentiable and Lipschitz. Consequently, also the product of continuously differentiable functions $\nabla^2 f(\boldsymbol{\theta}^k)^{-1} w^{k+1}$ is continuously differentiable and locally Lipschitz. The above facts guarantee that the functions φ , ϕ , and Ψ are continuously differentiable and Lipschitz.

The boundary-layer system $\xi^k = \varphi(\xi^{k-1}, \bar{\boldsymbol{\theta}})$, obtained by setting $\beta = 0$, reaches the steady-state value $\xi^*(\bar{\boldsymbol{\theta}}) := \{g^* = w^* = \nabla f(\bar{\boldsymbol{\theta}}), \tilde{\boldsymbol{\theta}}^* = \mathbf{0}\}$ for any choice of $\bar{\boldsymbol{\theta}}$. Recalling that $\boldsymbol{\theta}^* = \mathbf{0}$, by the first-order optimality condition $\nabla f(\mathbf{0}) = \mathbf{0}$, which implies $\phi(\mathbf{0}, \xi^*(\mathbf{0})) = \mathbf{0}$ and $\Psi(y^*(\mathbf{0}), \mathbf{0}) = \mathbf{0}$. Finally, a suitable choice for the Lyapunov function is $V(\bar{\boldsymbol{\theta}}) = f(\bar{\boldsymbol{\theta}})$. Indeed, using Taylor expansion, the fact that $f(\mathbf{0}) = \mathbf{0}$ and Assumption 4.1 the following inequalities hold.

$$\frac{\mu^2}{2} \|\bar{\boldsymbol{\theta}}\|^2 \leq V(\bar{\boldsymbol{\theta}}) \leq \frac{L^2}{2} \|\bar{\boldsymbol{\theta}}\|^2, \quad \left\| \frac{\partial V}{\partial \bar{\boldsymbol{\theta}}} \right\| \leq L \|\bar{\boldsymbol{\theta}}\|,$$

Since all the required conditions are satisfied,

$$\begin{aligned}
\frac{\partial V}{\partial \bar{\boldsymbol{\theta}}} \phi(\bar{\boldsymbol{\theta}}, \xi^*(\bar{\boldsymbol{\theta}})) &= -\nabla f(\bar{\boldsymbol{\theta}})^T \left(\frac{1}{N} \sum_{i=1}^N [\nabla^2 f_i(\bar{\boldsymbol{\theta}})^{-1}] \right) \nabla f(\bar{\boldsymbol{\theta}}) \\
&\leq -\frac{1}{L} \|\nabla f(\bar{\boldsymbol{\theta}})\|^2 \leq -\frac{\mu^2}{L} \|\bar{\boldsymbol{\theta}}\|^2.
\end{aligned}$$

we can invoke Theorem 4.4.1 to complete the proof.

□

Remark 4.2. *The proof of Theorem 4.4.2 utilizes the time-scales separation principle, which allows us to prove linear convergence and get concise proofs. However, this principle is primarily an existential theorem, making it difficult to determine the exact convergence rate coefficient. While we do not explicitly compare our theoretical convergence rate with other algorithms, it is important to note that such a rate can be at most linear for all the consensus-based algorithms due to the bottleneck imposed by the mixing matrices.*

4.4.2 NEC-GIANT

The key point in the associated convergence analysis is that each iteration of NEC-GIANT is equivalent to one of GIANT, and therefore, the convergence trajectories of the two algorithms are identical. The only difference between the two algorithms is the way the averages are computed: while GIANT requires a central aggregator server, NEC-GIANT achieves the same result in a fully distributed manner. This is formalized by the following proposition.

Proposition 1. *At the end of each iteration, the quantities computed by NEC-GIANT and GIANT (the global gradient, the global approximate Newton direction, and the updated decision vector) are identical, provided that both algorithms employ the same step size.*

Proof. The proof follows from the following facts: (i) all nodes are initialized with the same arbitrary θ_i^0 and perform the same updates in parallel, (ii) NEC-GIANT follows the same steps as GIANT, computing the same intermediate quantities and applying the same update rule, (iii) the FTDC protocol returns the exact average of the local quantities. □

As a consequence, in Algorithm 7 it holds $g_i^k = g_1^k$, $\eta_i^k = \eta_1^k$ and $\theta_i^k = \theta_1^k$ for all nodes $i \in N$ and for all iterations $k \in \mathbb{N}$. NEC-GIANT also retains the linear-quadratic local convergence property of GIANT, in particular, for problem (4.1), with a communication overhead given as follows:

Proposition 2. *Each iteration of NEC-GIANT involves $2(D-1)$ additional transmissions per node compared to GIANT, where $D+1 \leq N$ is the number of distinct eigenvalues of the consensus matrix P .*

The above result follows from the following facts: (i) in GIANT, computing an average at the central server and broadcasting the result involves two transmissions per node, (ii) each call to the FTDC routine using the matrix P involves $D+1$ transmissions per agent, (iii) two averages per iteration are computed in both algorithms. We remark that the additional communications are typically short-range as they cross a single edge of the network, unlike in federated learning, where all clients have to reach a possibly far central server.

4.5. EXPERIMENTAL RESULTS

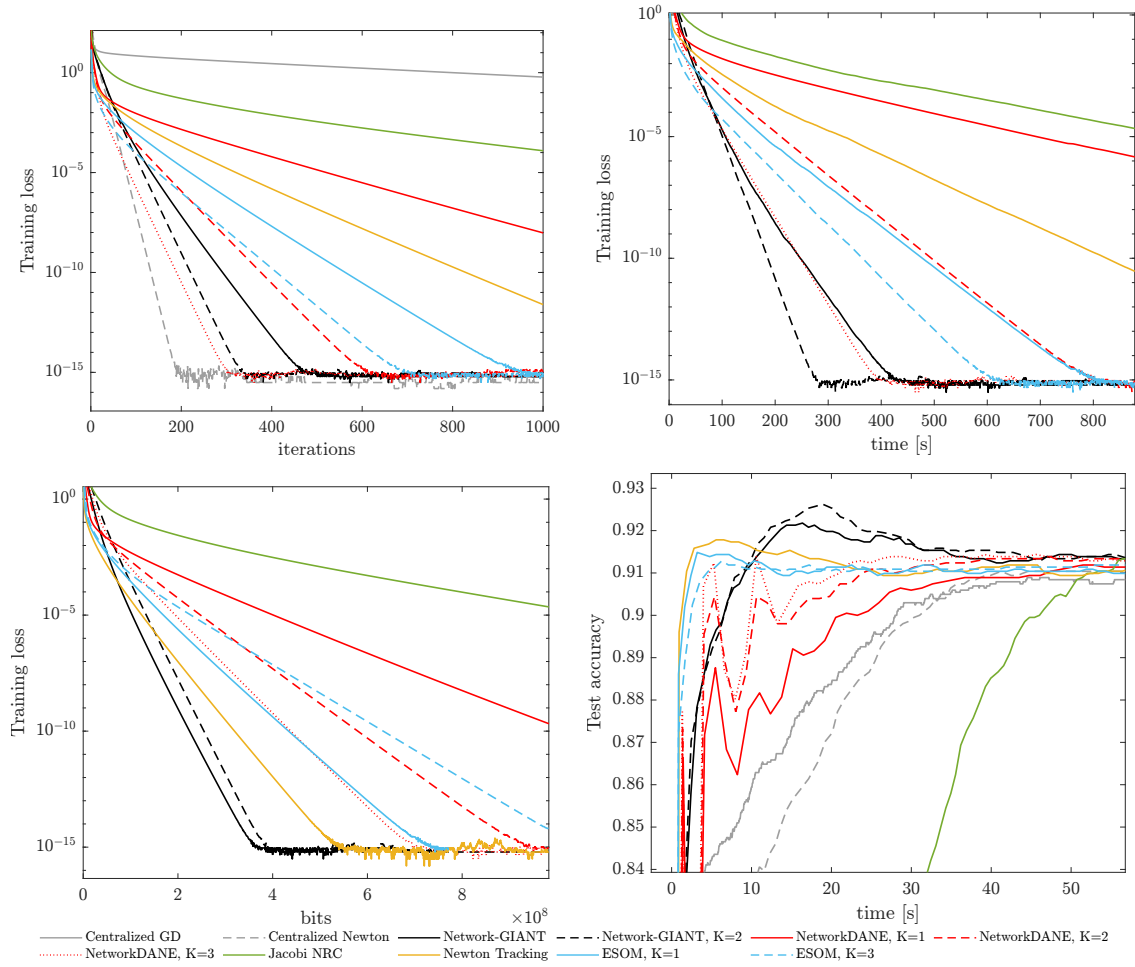


Figure 4.1: The first three figures from the left show the convergence performance on the training set, while the last figure displays the transient evolution of the accuracy on the test set of the MNIST dataset. The convergence is plotted against multiple metrics, namely the number of iterations, the overall computation time, and the overall communication cost in bits of the network.

4.5 Experimental Results

In this section, we practically assess the performance of Network-GIANT and Network-GIANT through numerical experiments and compare them with several other state-of-the-art distributed algorithms. In particular, we perform distributed binary classification between two classes of the well-known datasets MNIST LeCun et al. [2010] and Covertype Dheeru and Taniskidou [2017] via logistic regression. For the MNIST dataset, we use Principal Component Analysis to reduce the feature size to $d = 300$, while for the Covertype dataset, we keep the original dimension $d = 54$. We consider a network of 20 nodes and build the mixing matrix P using the Metropolis-Hastings weights Xiao et al.

4.5. EXPERIMENTAL RESULTS

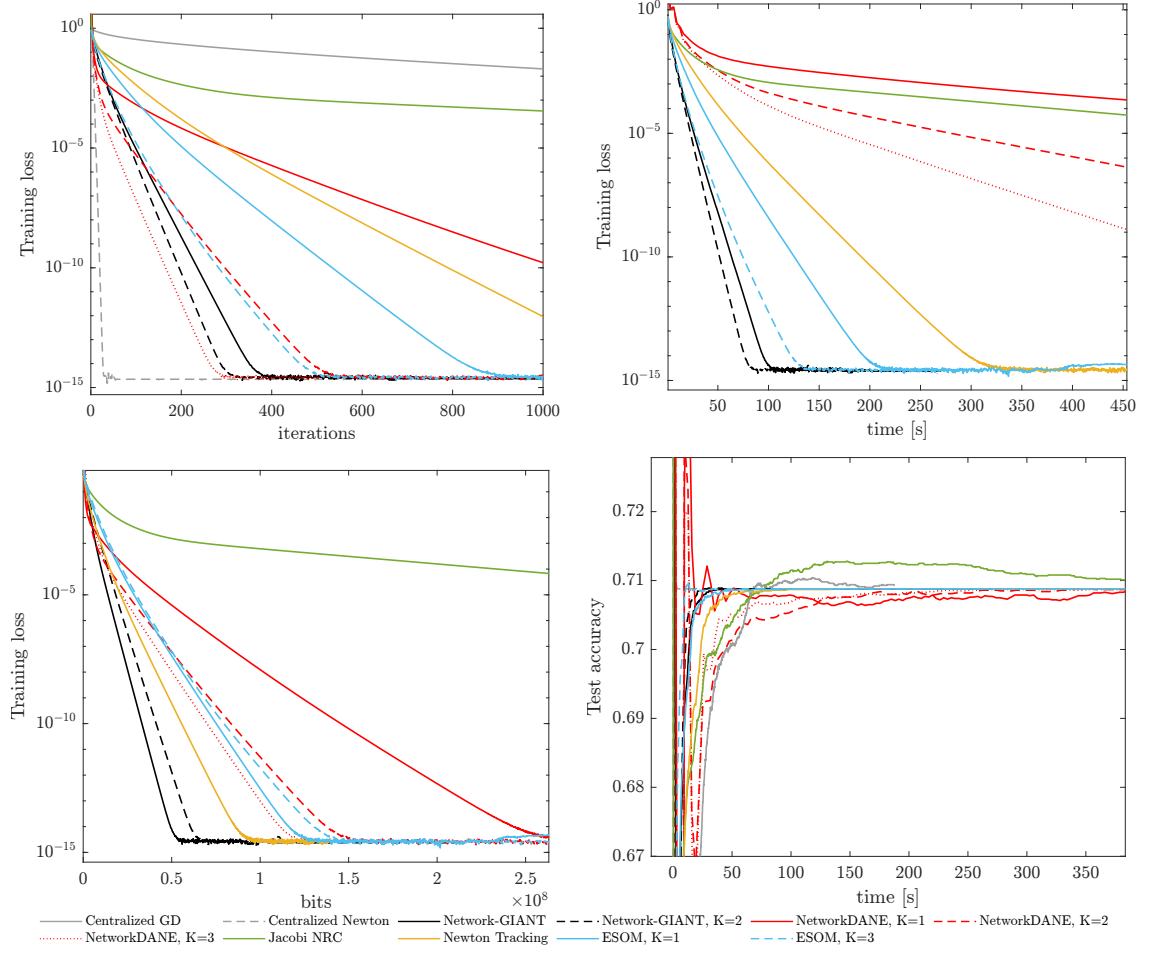


Figure 4.2: The first three figures from the left show the convergence performance on the training set, while the last figure displays the transient evolution of the accuracy on the test set of the Covertypе dataset. The convergence is plotted against multiple metrics, namely the number of iterations, the overall computation time, and the overall communication cost in bits of the network.

[2007]. Each node i is given m_i data samples $(x^{(i)}, y^{(i)}) \in \mathbb{R}^d \times \{-1, +1\}$, where the first element in the tuple is the predictor and the second is the target response. The local cost functions are the regularized log-losses

$$f_i(\boldsymbol{\theta}) = \frac{1}{m_i} \sum_{j=1}^{m_i} \log(1 + e^{-y_j^{(i)}(\boldsymbol{\theta}^T x_j^{(i)})}) + \frac{\lambda}{2} \|\boldsymbol{\theta}\|^2, \quad (4.15)$$

while the training loss shown in the plots is computed as $[\frac{1}{N} \sum_{i=1}^N f_i(\boldsymbol{\theta}_i) - f(\boldsymbol{\theta}^*)] / f(\boldsymbol{\theta}^*)$.

We test Network-GIANT both in its vanilla version and when $K = 2$ consecutive consensus steps on $\{u_i\}_{i=1}^N$ are performed. We compare them with the following fully

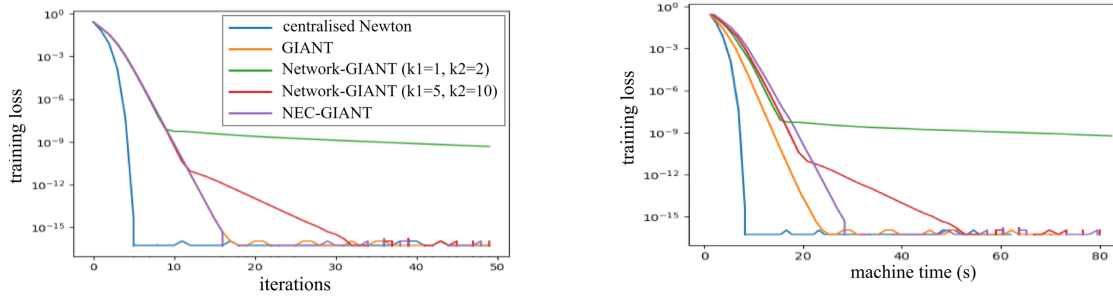
distributed algorithms: Network-DANE (Li et al. [2020e]) and ESOM- K (Mokhtari et al. [2016b]), both with different values K of mixing rounds; NRC (Varagnolo et al. [2015]) in its Jacobi version, which keeps the communication cost linear in d ; Newton Tracking (Zhang et al. [2021]). We do not compare Network-GIANT to any federated learning algorithm, not even GIANT, because the results would depend on the chosen network topology. Also, we do not consider the inexact algorithms Network Newton (Mokhtari et al. [2016a]) and DQN (Bajovic et al. [2017]), which only converge to a neighborhood of the solution. Rather, we include the standard centralized gradient descent (GD) with backtracking line search and the centralized Newton-Raphson method as a reference in the comparisons. For all the algorithms, we tune the parameters as suggested in the respective papers, selecting the best performance values.

Fig. 4.1 and 4.2 provide interesting insights into the strengths and weaknesses of the algorithms under examination, allowing a complete performance comparison that takes into account all the main evaluation metrics on two different datasets. When the convergence rate is expressed in terms of either run time or communication cost, Network-GIANT is superior to all its competitors. This consistency does not hold for the other algorithms: for example, ESOM outperforms Newton Tracking in terms of the number of iterations and computation time, but the opposite is true if we look at the communication cost. When looking at the number of iterations, Network-DANE with $K = 3$ converges slightly faster than the others. However, each iteration of Network-DANE involves an inner minimization problem, which, in general, is more computationally demanding compared to the other algorithms, even for d in the size of hundreds. Finally, the relatively poor performance of Jacobi NRC can be attributed to the fact that this algorithm exploits only the diagonal of local Hessian matrices, leading to slow convergence for skewed objective functions.

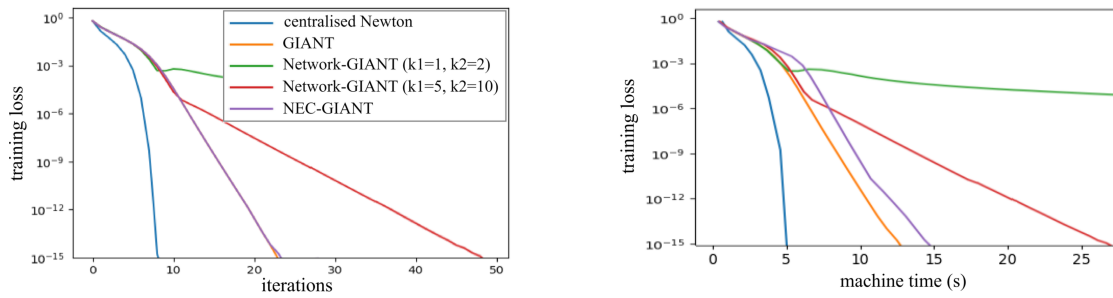
The numerical experiments also reveal the importance of the parameter K , which dramatically affects the ranking of the various algorithms. In particular, for Network-DANE, bigger values of K always seem beneficial concerning all metrics up to a certain threshold. Instead, for Network-GIANT and ESOM- K , we observe a trade-off: higher values of K decrease both the number of iterations and the run-time needed for convergence, but simultaneously increase the total number of bits transmitted between the nodes.

For completeness, we also show the performance on the test set, recalling that generalization to out-of-sample data is a bigger problem outside the scope of this work. All algorithms, with the exception of Jacobi NRC, achieve similar accuracy at a comparable rate, exhibiting oscillations that may be due to the statistical difference between training and test sets.

Fig. 4.3 shows the performance comparison of the NEC-GIANT algorithm with the centralised Newton-Raphson method, GIANT, and Network-GIANT. As expected, the



(a) Results on the dataset Covertypes Dheeru and Taniskidou [2017].



(b) Results on the dataset MNIST LeCun et al. [2010].

Figure 4.3: The figure shows a convergence comparison of centralised Newton, GIANT, and Network-GIANT, where k_1 indicates consensus rounds to determine global gradient, w_i^{k+1} , and k_2 indicates consensus rounds to determine global parameter vector, θ_i^{k+1} , and NEC-GIANT. In both the figures, the first plot shows training loss vs iteration, and the second plot shows training loss vs machine time in seconds.

convergence trajectory of NEC-GIANT is identical to that of GIANT up to numerical precision in terms of the number of iterations, and the additional communications due to FTDC only increase the machine runtime to a limited extent. The convergence rate of Network-GIANT is initially very similar to that of GIANT and NEC-GIANT but shows a slowdown once a certain error is reached. In summary, NEC-GIANT efficiently circumvents the absence of a central server and shows superior performance compared to the Network-GIANT.

4.6 Conclusion

This chapter presents the development of highly efficient network learning algorithms named Network-GIANT and NEC-GIANT. The proposed algorithms eliminate the need for a parameter server by applying the GIANT algorithm to a network learning framework. It leverages the concepts of average consensus and gradient tracking to achieve an approximate Newton-type peer-to-peer distributed optimization in Network-GIANT

and finite-time exact consensus to mimic the convergence properties of GIANT in the network settings. The convergence analysis of Network-GIANT indicates a semi-global exponential convergence to the exact solution. The chapter provides an empirical comparison of Network-GIANT with other state-of-the-art network learning algorithms, such as Network-DANE, Jacobi NRC, Newton Tracking, and ESOM. The results demonstrate that Network-GIANT outperforms its competitors in terms of convergence, communication cost, and computation performance. We empirically compared NEC-GIANT with GIANT and Network-GIANT to show improved convergence performance of NEC-GIANT over Network-GIANT. We foresee interesting possibilities for future work, such as analyzing robustness to disturbances and noise injection and introducing quantization to reduce communication costs further. We also plan to explore extensions to the non-convex case, for example, following a trust-region approach or using cubic regularization.

5

Conclusions and Future Work

Our research introduced a novel distributed machine learning algorithm called DANDA, which integrates determinantal averaging with a low-rank Hessian approximation at each node. This approach, along with over-the-air edge learning, offers the advantage of rapid convergence while utilizing approximation methods to lower the computation cost per iteration. DANDA presents a novel approach to addressing the inversion bias problem in distributed Newton's method. By introducing determinantal averaging, the proposed method effectively weights local estimates based on the local Hessian determinant, leading to a more accurate global approximation. The algorithm achieves significantly faster convergence than traditional gradient-based approaches by employing a Newton-type method, thereby reducing communication costs due to the fewer iterations needed for training. The algorithm's innovative use of analog over-the-air MAC communication offers substantial benefits over conventional digital or orthogonal MACs, leveraging over-the-air signal superposition to make bandwidth requirements independent of the number of nodes.

Our experimental findings reveal that DANDA achieves significantly faster convergence than the state-of-the-art gradient-based GBMA. Furthermore, it demonstrates favorable performance compared to true Hessian-based LN in terms of total power consumption and computational time, all while providing enhanced privacy assurance by not transmitting the parameter vector, θ itself.

Expanding upon our work, we introduced two efficient distributed learning algorithms: Lazy-DANDA and Lazy-DANTA. These algorithms optimize communication and computation efficiency by incorporating adaptive Hessian updates and approximate Hessian evaluations, thereby minimizing unnecessary computations. To maximize bandwidth usage, we adopt an analog over-the-air transmission communication strategy. Additionally, we enhance the algorithm's energy efficiency by integrating the channel inversion technique based on the channel gain threshold. Empirical evaluations demonstrate the effectiveness of our algorithms when compared to other state-of-the-art distributed learning

approaches.

Our work further presents the development of Network-GIANT, a highly efficient network learning algorithm. Network-GIANT eliminates the need for a parameter server by applying the GIANT algorithm within a network learning framework. Network-GIANT achieves approximate Newton-type peer-to-peer distributed optimization by leveraging average consensus and gradient tracking. The convergence analysis indicates semi-global exponential convergence to the exact solution. Empirical comparisons with other network learning algorithms, including Network-DANE, Jacobi NRC, Newton Tracking, and ESOM, underscore Network-GIANT’s convergence, communication cost, and computation performance superiority.

We also introduced the NEC-GIANT, which extends to the fully distributed scenario of the federated algorithm GIANT. Unlike the Network-GIANT, which is also a fully distributed algorithm inspired by GIANT, NEC-GIANT preserves GIANT’s convergence properties. The peculiarity of NEC-GIANT is the use of finite-time distributed consensus, which removes dependence on the central server without any performance degradation, effectively bridging the gap between federated and fully distributed scenarios. Numerical tests confirm the superior convergence performance of the proposed algorithm.

While this research has significantly contributed to distributed learning, several promising avenues exist for future exploration. Extending the analysis of DANDA from federated to fully distributed learning would provide a more comprehensive understanding of its applicability in diverse network topologies and communication constraints. Investigating the theoretical convergence properties of DANDA under the influence of channel fading effects would enhance its robustness in real-world wireless communication environments. Implementing enhanced communication and rapid average consensus among nodes in fully distributed learning, drawing from the methodologies outlined in Ying et al. [2021], could further improve the efficiency and accuracy of the algorithm. Exploring fully distributed learning utilizing zeroth-order optimization techniques would enable network optimization to be applied to scenarios where gradient information is unavailable or costly.

While this work relied on manual hyperparameter tuning strategies and automating this could be highly time consuming, future research could explore learning the optimal hyperparameters directly through meta-learning approaches. Such methods aim to “learn to learn,” enabling models to adapt hyperparameters Andrychowicz et al. [2016].

It should be noted that the experiments were performed entirely in simulation, without modeling network-level effects such as latency or device failures. Future work will involve testing on real-world edge networks to evaluate performance under practical deployment conditions.

By pursuing these research directions, we can continue to advance the state-of-the-art in distributed learning and unlock new possibilities for collaborative intelligence in

privacy-sensitive and resource-constrained environments.

6

Appendix A

6.1 Proof of Theorem 3.4.2

Proof. We begin our proof by considering the second-order Taylor's series equality Boyd and Vandenberghe [2004] using the strong convexity assumption of the global objective function, $F(\boldsymbol{\theta}_k)$ for the k^{th} iteration. For $\boldsymbol{\theta}_k, \boldsymbol{\theta} + \beta\Delta\boldsymbol{\theta}_{nt} \in \Theta$ we have,

$$F(\boldsymbol{\theta}_k + \beta\Delta\boldsymbol{\theta}_{nt}) = F(\boldsymbol{\theta}_k) + \beta\nabla F(\boldsymbol{\theta}_k)^T \Delta\boldsymbol{\theta}_{nt} + \frac{1}{2}\beta^2 \Delta\boldsymbol{\theta}_{nt}^T \nabla^2 F(\boldsymbol{\theta}') \Delta\boldsymbol{\theta}_{nt} \quad (6.1)$$

for some $\boldsymbol{\theta}'$ on the line segment $[\boldsymbol{\theta}_k, \boldsymbol{\theta}_k + \beta\Delta\boldsymbol{\theta}_{nt}]$

$$F(\boldsymbol{\theta}_k + \beta\Delta\boldsymbol{\theta}_{nt}) \leq F(\boldsymbol{\theta}_k) + \beta\nabla F(\boldsymbol{\theta}_k)^T \Delta\boldsymbol{\theta}_{nt} + \frac{1}{2}\beta^2 M \|\Delta\boldsymbol{\theta}_{nt}\|_2^2 \quad (\text{Assumption 3.3})$$

Taking the expectation over the channel effect gives,

$$\begin{aligned} \mathbb{E}[F(\boldsymbol{\theta}_k + \beta\Delta\boldsymbol{\theta}_{nt})] &\leq \mathbb{E}[F(\boldsymbol{\theta}_k) + \beta\nabla F(\boldsymbol{\theta}_k)^T \Delta\boldsymbol{\theta}_{nt} + \frac{1}{2}\beta^2 M \|\Delta\boldsymbol{\theta}_{nt}\|_2^2] \\ &= \mathbb{E}[F(\boldsymbol{\theta}_k)] + \mathbb{E}[\beta\nabla F(\boldsymbol{\theta}_k)^T \Delta\boldsymbol{\theta}_{nt}] + \mathbb{E}[\frac{1}{2}\beta^2 M \|\Delta\boldsymbol{\theta}_{nt}\|_2^2] \\ &= \mathbb{E}[F(\boldsymbol{\theta}_k)] + \beta\mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \Delta\boldsymbol{\theta}_{nt}] + \frac{1}{2}\beta^2 M \mathbb{E}[\|\Delta\boldsymbol{\theta}_{nt}\|_2^2] \end{aligned} \quad (6.2)$$

Before proceeding further, we first evaluate the terms $\mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \Delta\boldsymbol{\theta}_{nt}]$ and $\mathbb{E}[\|\Delta\boldsymbol{\theta}_{nt}\|_2^2]$ where the expectation is over the random additive noise and the channel gain effect.

The expected value of $\|\Delta\boldsymbol{\theta}_{nt}\|_2^2$, can be expanded as

$$\mathbb{E}[\|\Delta\boldsymbol{\theta}_{nt}\|_2^2] = \mathbb{E}\left[\left\| - \left(\frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \mathbf{H}_{s_n}^{-1} \nabla F(\boldsymbol{\theta}_k)}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n + \tilde{w}_k^2} + \frac{\tilde{\mathbf{w}}_k^1}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n + \tilde{w}_k^2} \right) \right\|^2\right] \quad \text{ref. (3.10)}$$

The subsequent analysis is based on regime of the asymptotic limit where the number of users/nodes N goes to ∞ . We ignore the term \tilde{w}_k^2 in the denominator from here on to simplify the analysis. Considering the fact that its mean is 0 and its variance is inversely proportional to D_k^2 (3.9) and hence, goes to 0 almost surely as $N \rightarrow \infty$.

$$\begin{aligned} \mathbb{E}[\|\Delta\boldsymbol{\theta}_{nt}\|_2^2] &= \mathbb{E}\left[\left\| - \left(\frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \mathbf{H}_{s_n}^{-1} \nabla F(\boldsymbol{\theta}_k)}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} + \frac{\tilde{\mathbf{w}}_k^1}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} \right) \right\|^2\right] \\ &= \mathbb{E}\left[\left\| \frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \mathbf{H}_{s_n}^{-1} \nabla F(\boldsymbol{\theta}_k)}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} \right\|^2\right] + \mathbb{E}\left[\left\| \frac{\tilde{\mathbf{w}}_k^1}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} \right\|^2\right] + \\ &\quad \mathbb{E}\left[\left(\frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \mathbf{H}_{s_n}^{-1} \nabla F(\boldsymbol{\theta}_k)}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} \right)^T \left(\frac{\tilde{\mathbf{w}}_k^1}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} \right)\right] \\ &= \mathbb{E}\left[\left\| \frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \mathbf{H}_{s_n}^{-1} \nabla F(\boldsymbol{\theta}_k)}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} \right\|^2\right] + \mathbb{E}\left[\left\| \frac{\tilde{\mathbf{w}}_k^1}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} \right\|^2\right] + 0 \quad (\mathbb{E}(\tilde{\mathbf{w}}_k^1) = 0) \\ &= \mathbb{E}\left[\left\| \frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \mathbf{H}_{s_n}^{-1} \nabla F(\boldsymbol{\theta}_k)}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} \right\|^2\right] + \mathbb{E}\left[\frac{d\sigma_w^2}{\alpha P_k D_k^2} \cdot \frac{1}{\left(\frac{1}{D_k} \sum_{n=1}^{D_k} a_n\right)^2}\right] \\ &= \mathbb{E}\left[\left\| \mathbf{H}^{-\frac{1}{2}} \mathbf{H}^{\frac{1}{2}} \frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \mathbf{H}_{s_n}^{-1} \nabla F(\boldsymbol{\theta}_k)}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} \mathbf{H}^{\frac{1}{2}} \mathbf{H}^{-\frac{1}{2}} \nabla F(\boldsymbol{\theta}_k) \right\|^2\right] + \mathbb{E}\left[\frac{d\sigma_w^2}{\alpha P_k D_k^2} \cdot \frac{1}{\left(\frac{1}{D_k} \sum_{n=1}^{D_k} a_n\right)^2}\right] \\ &\leq \mathbb{E}[\|\mathbf{H}^{-\frac{1}{2}}\|^2] \cdot \mathbb{E}\left[\left\| \mathbf{H}^{\frac{1}{2}} \frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \frac{1}{1+\epsilon_0} \mathbf{H}_n^{-1}}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n} \mathbf{H}^{\frac{1}{2}} \right\|^2\right] \cdot \mathbb{E}[\|\mathbf{H}^{-\frac{1}{2}} \nabla F(\boldsymbol{\theta}_k)\|^2] + \\ &\quad \mathbb{E}\left[\frac{d\sigma_w^2}{\alpha P_k D_k^2} \cdot \frac{1}{\left(\frac{1}{D_k} \sum_{n=1}^{D_k} a_n\right)^2}\right] \\ &\leq \mathbb{E}[\|\mathbf{H}^{-1}\|] \cdot \mathbb{E}\left[\left\| \mathbf{H}^{\frac{1}{2}} \left(1 + \frac{\eta}{\sqrt{D_k}}\right) \frac{1}{1+\epsilon_0} \mathbf{H}^{-1} \mathbf{H}^{\frac{1}{2}} \right\|^2\right] \cdot \mathbb{E}[\|\mathbf{H}^{-\frac{1}{2}} \nabla F(\boldsymbol{\theta}_k)\|^2] + \\ &\quad \mathbb{E}\left[\frac{d\sigma_w^2}{\alpha P_k D_k^2} \cdot \frac{1}{\left(\frac{1}{D_k} \sum_{n=1}^{D_k} a_n\right)^2}\right] \end{aligned}$$

$$\leq \frac{1}{m(1+\epsilon_0)^2} \cdot \left(1 + \frac{\eta}{\sqrt{D_k}}\right)^2 \cdot \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] + \mathbb{E}\left[\frac{d\sigma_w^2}{\alpha P_k D_k^2} \cdot \frac{1}{\left(\frac{1}{D_k} \sum_{n=1}^{D_k} a_n\right)^2}\right]$$

($m \leq \|H\|$ Assumption 3.3)

$$\leq \frac{1}{m(1+\epsilon_0)^2} \cdot \left(1 + \frac{\eta}{\sqrt{D_k}}\right)^2 \cdot \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] + \mathbb{E}\left[\frac{1}{D_k^2} \cdot \frac{d\sigma_w^2}{\alpha P_k \lambda_l^{2d}}\right]$$

($\sum_{n=1}^{D_k} a_n \geq D_k \lambda_l^d$, where λ_l is the lowest eigenvalue of the local Hessians among nodes)

$$\mathbb{E}[\|\Delta\boldsymbol{\theta}_{nt}\|^2] \leq \frac{1}{m(1+\epsilon_0)^2} \cdot \left(1 + \frac{\eta}{\sqrt{D_k}}\right)^2 \cdot \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] + \frac{d\sigma_w^2}{D_k^2 \alpha P_k} \cdot \frac{1}{\lambda_l^{2d}} \quad (6.3)$$

Now, we evaluate the value of $\mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \Delta\boldsymbol{\theta}_{nt}]$

$$\begin{aligned} \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \Delta\boldsymbol{\theta}_{nt}] &= \mathbb{E}\left[-\nabla F(\boldsymbol{\theta}_k)^T \frac{\mathbf{v}_k^1}{v_k^2}\right] \\ &= -\mathbb{E}\left[\nabla F(\boldsymbol{\theta}_k)^T \frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \mathbf{H}_{s_n}^{-1} \nabla F(\boldsymbol{\theta}_k) + \tilde{\mathbf{w}}_k^1}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n}\right] \quad \text{ref. (3.10)} \\ &\leq -\mathbb{E}\left[\nabla F(\boldsymbol{\theta}_k)^T \frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \frac{1}{1-\epsilon_0} \mathbf{H}_n^{-1} \nabla F(\boldsymbol{\theta}_k) + \tilde{\mathbf{w}}_k^1}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n}\right] \\ &\leq -\frac{1}{1-\epsilon_0} \mathbb{E}\left[\nabla F(\boldsymbol{\theta}_k)^T \frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \mathbf{H}_n^{-1} \nabla F(\boldsymbol{\theta}_k)}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n}\right] - \mathbb{E}\left[\nabla F(\boldsymbol{\theta}_k)^T \frac{\tilde{\mathbf{w}}_k^1}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n}\right] \\ &= -\frac{1}{1-\epsilon_0} \mathbb{E}\left[\nabla F(\boldsymbol{\theta}_k)^T \frac{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n \mathbf{H}_n^{-1} \nabla F(\boldsymbol{\theta}_k)}{\frac{1}{D_k} \sum_{n=1}^{D_k} a_n}\right] \quad (\mathbb{E}(\tilde{\mathbf{w}}_k^1) = 0) \\ &\leq -\frac{1}{1-\epsilon_0} \left(1 - \frac{\eta}{\sqrt{D_k}}\right) \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] \quad \text{ref. Theorem 3.4.1} \end{aligned}$$

$$\mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \Delta\boldsymbol{\theta}_{nt}] \leq -\frac{1}{1-\epsilon_0} \left(1 - \frac{\eta}{\sqrt{D_k}}\right) \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] \quad (6.4)$$

By substituting (6.4) and (6.3) in (6.2), we get

$$\begin{aligned}
\mathbb{E}[F(\boldsymbol{\theta}_k + \beta\Delta\boldsymbol{\theta}_{nt})] &\leq \mathbb{E}[F(\boldsymbol{\theta}_k)] - \beta \frac{1}{1 - \epsilon_0} \left(1 - \frac{\eta}{\sqrt{D_k}}\right) \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] + \\
&\quad \frac{1}{2} \beta^2 M \frac{1}{m(1 + \epsilon_0)^2} \cdot \left(1 + \frac{\eta}{\sqrt{D_k}}\right)^2 \cdot \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] + \\
&\quad \frac{d\sigma_w^2}{D_k^2 \alpha P_k} \cdot \frac{1}{\lambda_l^{2d}} \\
\mathbb{E}[F(\boldsymbol{\theta}_k + \beta\Delta\boldsymbol{\theta}_{nt})] - \mathbb{E}[F(\boldsymbol{\theta}_k)] &\leq -\beta \left(\frac{1}{1 - \epsilon_0} \left(1 - \frac{\eta}{\sqrt{D_k}}\right) \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] + \right. \\
&\quad \left. \frac{1}{2} \beta M \frac{1}{m(1 + \epsilon_0)^2} \cdot \left(1 + \frac{\eta}{\sqrt{D_k}}\right)^2 \cdot \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] \right) + \\
&\quad \frac{d\sigma_w^2}{D_k^2 \alpha P_k} \cdot \frac{1}{\lambda_l^{2d}}
\end{aligned}$$

The algorithm converges if the right-hand side of the equation is negative. For that, the first term on the right-hand side should be greater than the sum of the second and third terms on the right-hand side of the above equation. To this end, the term $\frac{d\sigma_w^2}{D_k^2 \alpha P_k} \cdot \frac{1}{\lambda_l^{2d}}$ can be neglected in comparison with other terms as it reaches 0 as $N \rightarrow \infty$. In other words, for a sufficiently large $N > N'$, we can simplify the above inequality as

$$\begin{aligned}
\mathbb{E}[F(\boldsymbol{\theta}_k + \beta\Delta\boldsymbol{\theta}_{nt})] - \mathbb{E}[F(\boldsymbol{\theta}_k)] &\leq -\beta \left(\frac{1}{1 - \epsilon_0} \left(1 - \frac{\eta}{\sqrt{D_k}}\right) \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] + \right. \\
&\quad \left. \frac{1}{2} \beta M \frac{1}{m(1 + \epsilon_0)^2} \cdot \left(1 + \frac{\eta}{\sqrt{D_k}}\right)^2 \cdot \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] \right)
\end{aligned}$$

It then follows that the right-hand side of the above is negative for a sufficiently large N , when

$$\begin{aligned}
&\beta \frac{1}{1 - \epsilon_0} \left(1 - \frac{\eta}{\sqrt{D_k}}\right) \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] \\
&> \frac{1}{2} \beta^2 M \left(\frac{1}{m(1 + \epsilon_0)^2} \cdot \left(1 + \frac{\eta}{\sqrt{D_k}}\right)^2 \cdot \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] \right)
\end{aligned}$$

Or,

$$\begin{aligned} \beta &< \frac{2 \frac{1}{1-\epsilon_0} \left(\left(1 - \frac{\eta}{\sqrt{D_k}}\right) \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] \right)}{M \left(\frac{1}{m(1+\epsilon_0)^2} \cdot \left(1 + \frac{\eta}{\sqrt{D_k}}\right)^2 \cdot \mathbb{E}[\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k)] \right)} \\ &= \frac{2 \left(\frac{1}{1-\epsilon_0} \left(1 - \frac{\eta}{\sqrt{D_k}}\right) \right)}{M \left(\frac{1}{m(1+\epsilon_0)^2} \left(1 + \frac{\eta}{\sqrt{D_k}}\right)^2 \right)} \end{aligned}$$

where the last inequality follows from the fact that $\nabla F(\boldsymbol{\theta}_k)^T \mathbf{H}^{-1} \nabla F(\boldsymbol{\theta}_k) \geq \frac{\epsilon^2}{m}$ as long as $\|\nabla F(\boldsymbol{\theta}_k)\| > \epsilon$ as required by the convergence criterion, i.e., the algorithm converges when $\|\nabla F(\boldsymbol{\theta}_k)\| \leq \epsilon$.

Finally, since the right-hand side of the above upper bound on β is an increasing function of D_k , one can conclude that, as $N \rightarrow \infty$, there is a sufficiently large $N' < N$, such that $D_k > N'$, $\forall k$ with high probability, and the convergence criterion of β can be written as

$$\beta < 2 \frac{m}{M} \left(\frac{(1 + \epsilon_0)^2 \left(1 - \frac{\eta}{\sqrt{N'}}\right)}{(1 - \epsilon_0) \left(1 + \frac{\eta}{\sqrt{N'}}\right)^2} \right) \tag{6.5}$$

□

Bibliography

- Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agueray Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017a.
- Naifu Zhang and Meixia Tao. Gradient statistics aware power control for over-the-air federated learning. *IEEE Transactions on Wireless Communications*, 20(8):5115–5128, 2021.
- Mohammad Mohammadi Amiri and Deniz Gündüz. Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air. *IEEE Transactions on Signal Processing*, 68:2155–2169, 2020a.
- Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K Leung, Christian Makaya, Ting He, and Kevin Chan. Adaptive federated learning in resource constrained edge computing systems. *IEEE journal on selected areas in communications*, 37(6):1205–1221, 2019.
- Mohammad Mohammadi Amiri and Deniz Gündüz. Federated learning over wireless fading channels. *IEEE Transactions on Wireless Communications*, 19(5):3546–3557, 2020b.
- Li Li, Yuxi Fan, Mike Tse, and Kuo-Yi Lin. A review of applications in federated learning. *Computers & Industrial Engineering*, 149:106854, 2020a.
- Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. Federated machine learning: Concept and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19, 2019.
- Sundar Pichai. Privacy should not be a luxury good. *The New York Times*, 8:25, 2019.
- Walter De Brouwer. The federated future is ready for shipping, 2019.
- David Leroy, Alice Coucke, Thibaut Lavril, Thibault Gisselbrecht, and Joseph Dureau. Federated learning for keyword spotting. In *ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pages 6341–6345. IEEE, 2019.
- Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE signal processing magazine*, 37(3): 50–60, 2020b.

-
- Viraaji Mothukuri, Reza M Parizi, Seyedamin Pouriyeh, Yan Huang, Ali Dehghantanha, and Gautam Srivastava. A survey on security and privacy of federated learning. *Future Generation Computer Systems*, 115:619–640, 2021.
- Kang Wei, Jun Li, Ming Ding, Chuan Ma, Howard H Yang, Farhad Farokhi, Shi Jin, Tony QS Quek, and H Vincent Poor. Federated learning with differential privacy: Algorithms and performance analysis. *IEEE transactions on information forensics and security*, 15:3454–3469, 2020.
- Robin C Geyer, Tassilo Klein, and Moin Nabi. Differentially private federated learning: A client level perspective. *arXiv preprint arXiv:1712.07557*, 2017.
- Dan Alistarh, Demjan Grubic, Jerry Li, Ryota Tomioka, and Milan Vojnovic. Qsgd: Communication-efficient sgd via gradient quantization and encoding. *Advances in neural information processing systems*, 30, 2017.
- Frank Seide, Hao Fu, Jasha Droppo, Gang Li, and Dong Yu. 1-bit stochastic gradient descent and its application to data-parallel distributed training of speech dnns. In *Fifteenth annual conference of the international speech communication association*, 2014.
- Yujun Lin, Song Han, Huizi Mao, Yu Wang, and William J Dally. Deep gradient compression: Reducing the communication bandwidth for distributed training. *arXiv preprint arXiv:1712.01887*, 2017.
- Yuchen Zhang and Xiao Lin. Disco: Distributed optimization for self-concordant empirical loss. In *International conference on machine learning*, pages 362–370. PMLR, 2015.
- Ohad Shamir, Nati Srebro, and Tong Zhang. Communication-efficient distributed optimization using an approximate newton-type method. In *International conference on machine learning*, pages 1000–1008. PMLR, 2014.
- Shusen Wang, Fred Roosta, Peng Xu, and Michael W Mahoney. Giant: Globally improved approximate newton method for distributed optimization. *Advances in Neural Information Processing Systems*, 31, 2018.
- Peng Xu, Fred Roosta, and Michael W Mahoney. Second-order optimization for non-convex machine learning: An empirical study. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 199–207. SIAM, 2020.
- Sashank J Reddi, Jakub Konečný, Peter Richtárik, Barnabás Póczós, and Alex Smola. Aide: Fast and communication efficient distributed optimization. *arXiv preprint arXiv:1608.06879*, 2016.

- Martin Jaggi, Virginia Smith, Martin Takáč, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. *Advances in neural information processing systems*, 27, 2014.
- Vipul Gupta, Avishek Ghosh, Michał Dereziński, Rajiv Khanna, Kannan Ramchandran, and Michael W Mahoney. Localnewton: Reducing communication rounds for distributed learning. In *Uncertainty in artificial intelligence*, pages 632–642. PMLR, 2021.
- Wei Wen, Cong Xu, Feng Yan, Chunpeng Wu, Yandan Wang, Yiran Chen, and Hai Li. Terngrad: Ternary gradients to reduce communication in distributed deep learning. *Advances in neural information processing systems*, 30, 2017.
- Nikko Ström. Scalable distributed dnn training using commodity gpu cloud computing. 2015.
- Alham Fikri Aji and Kenneth Heafield. Sparse communication for distributed gradient descent. *arXiv preprint arXiv:1704.05021*, 2017.
- Chia-Yu Chen, Jungwook Choi, Daniel Brand, Ankur Agrawal, Wei Zhang, and Kailash Gopalakrishnan. Adacomp: Adaptive residual gradient compression for data-parallel distributed training. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- Cédric Renggli, Saleh Ashkboos, Mehdi Aghagolzadeh, Dan Alistarh, and Torsten Hoefler. Sparcml: High-performance sparse communication for machine learning. In *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–15, 2019.
- Peng Jiang and Gagan Agrawal. A linear speedup analysis of distributed deep learning with sparse and quantized communication. *Advances in Neural Information Processing Systems*, 31, 2018.
- Kai Yang, Tao Jiang, Yuanming Shi, and Zhi Ding. Federated learning via over-the-air computation. *IEEE transactions on wireless communications*, 19(3):2022–2035, 2020.
- Guangxu Zhu, Yong Wang, and Kaibin Huang. Broadband analog aggregation for low-latency federated edge learning. *IEEE Transactions on Wireless Communications*, 19(1):491–506, 2019.
- Mohammad Mohammadi Amiri and Deniz Gündüz. Over-the-air machine learning at the wireless edge. In *2019 IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, pages 1–5. IEEE, 2019.

- Tomer Sery and Kobi Cohen. On analog gradient descent learning over multiple access fading channels. *IEEE Transactions on Signal Processing*, 68:2897–2911, 2020.
- Dongzhu Liu and Osvaldo Simeone. Privacy for free: Wireless federated learning via uncoded transmission with adaptive power control. *IEEE Journal on Selected Areas in Communications*, 39(1):170–185, 2020.
- Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems*, 2:429–450, 2020c.
- Nicolò Dal Fabbro, Subhrakanti Dey, Michele Rossi, and Luca Schenato. Shed: A newton-type algorithm for federated learning based on incremental hessian eigenvector sharing. *arXiv e-prints*, pages arXiv–2202, 2022.
- Dimitri Bertsekas and John Tsitsiklis. *Parallel and distributed computation: numerical methods*. Athena Scientific, 2015.
- John N Tsitsiklis. *Problems in decentralized decision making and computation*. PhD thesis, Massachusetts Institute of Technology, 1984.
- John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.
- Angelia Nedic and Asuman Ozdaglar. Distributed subgradient methods for multi-agent optimization. *IEEE Transactions on Automatic Control*, 54(1):48–61, 2009.
- Dušan Jakovetić, Joao Xavier, and José MF Moura. Fast distributed gradient methods. *IEEE Transactions on Automatic Control*, 59(5):1131–1146, 2014.
- Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Transactions on Signal Processing*, 62(7):1750–1761, 2014.
- Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015a.
- Michael Zargham, Alejandro Ribeiro, Asuman Ozdaglar, and Ali Jadbabaie. Accelerated dual descent for network flow optimization. *IEEE Transactions on Automatic Control*, 59(4):905–920, 2013.

-
- Aryan Mokhtari, Qing Ling, and Alejandro Ribeiro. Network newton distributed optimization methods. *IEEE Transactions on Signal Processing*, 65(1):146–161, 2016a.
- Jiaqi Zhang, Keyou You, and Tamer Başar. Distributed adaptive newton methods with global superlinear convergence. *Automatica*, 138:110156, 2022.
- Jiaojiao Zhang, Qing Ling, and Anthony Man-Cho So. A newton tracking algorithm with exact linear convergence for decentralized consensus optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 7:346–358, 2021.
- Huayan Guo, An Liu, and Vincent KN Lau. Analog gradient aggregation for federated learning over wireless networks: Customized design and convergence analysis. *IEEE Internet of Things Journal*, 8(1):197–210, 2020.
- Tomer Sery, Nir Shlezinger, Kobi Cohen, and Yonina C Eldar. Over-the-air federated learning from heterogeneous data. *IEEE Transactions on Signal Processing*, 69:3796–3811, 2021.
- Canh T Dinh, Nguyen H Tran, Tuan Dung Nguyen, Wei Bao, Amir Rezaei Balef, Bing B Zhou, and Albert Y Zomaya. Done: Distributed approximate newton-type method for federated edge learning. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2648–2660, 2022.
- Sheng Hua, Kai Yang, and Yuanming Shi. On-device federated learning via second-order optimization with over-the-air computation. In *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pages 1–5. IEEE, 2019.
- Mounssif Krouka, Anis Elgabli, Chaouki Ben Issaid, and Mehdi Bennis. Communication-efficient federated learning: A second order newton-type method with analog over-the-air aggregation. *IEEE Transactions on Green Communications and Networking*, 6(3):1862–1874, 2022.
- Michal Dereziński and Michael W Mahoney. Distributed estimation of the inverse hessian by determinantal averaging. *Advances in Neural Information Processing Systems*, 32, 2019.
- Thierry Bertin-Mahieux, Daniel PW Ellis, Brian Whitman, and Paul Lamere. The million song dataset. 2011.
- Murat A Erdogdu and Andrea Montanari. Convergence rates of sub-sampled newton methods. *Advances in Neural Information Processing Systems*, 28, 2015.
- Dua Dheeru and E Karra Taniskidou. Uci machine learning repository. 2017.

- Mingzhe Chen, Deniz Gündüz, Kaibin Huang, Walid Saad, Mehdi Bennis, Aneta Vulgarakis Feljan, and H Vincent Poor. Distributed learning in wireless networks: Recent progress and future challenges. *IEEE Journal on Selected Areas in Communications*, 39(12):3579–3605, 2021a.
- Naman Agarwal, Brian Bullins, and Elad Hazan. Second-order stochastic optimization for machine learning in linear time. *Journal of Machine Learning Research*, 18(116): 1–40, 2017. URL <http://jmlr.org/papers/v18/16-491.html>.
- Richard H Byrd, Gillian M Chin, Jorge Nocedal, and Yuchen Wu. Sample size selection in optimization methods for machine learning. *Mathematical programming*, 134(1): 127–155, 2012.
- Farbod Roosta-Khorasani and Michael W Mahoney. Sub-sampled newton methods i: globally convergent algorithms. *arXiv preprint arXiv:1601.04737*, 2016.
- Mert Pilanci and Martin J Wainwright. Newton sketch: A near linear-time optimization algorithm with linear-quadratic convergence. *SIAM Journal on Optimization*, 27(1): 205–245, 2017.
- Peng Xu, Jiyan Yang, Fred Roosta, Christopher Ré, and Michael W Mahoney. Sub-sampled newton methods with non-uniform sampling. *Advances in neural information processing systems*, 29, 2016.
- Nikita Doikov, Martin Jaggi, et al. Second-order optimization with lazy hessians. In *International Conference on Machine Learning*, pages 8138–8161. PMLR, 2023.
- Xiaowen Cao, Guangxu Zhu, Jie Xu, and Kaibin Huang. Optimal power control for over-the-air computation. In *2019 IEEE Global Communications Conference (GLOBECOM)*, pages 1–6. IEEE, 2019.
- Ganesh Sharma and Subhrakanti Dey. On analog distributed approximate newton with determinantal averaging. In *2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–7, 2022a. doi: 10.1109/PIMRC54779.2022.9977466.
- Haishan Ye, Luo Luo, and Zhihua Zhang. Approximate newton methods. *The Journal of Machine Learning Research*, 22(1):3067–3107, 2021.
- William Feller. *An Introduction to Probability Theory and its Applications, Volume 1*. J. Wiley & Sons: New York, 1968.
- H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017b.

- Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. Federated learning: Challenges, methods, and future directions. *IEEE Signal Processing Magazine*, 37(3): 50–60, 2020d.
- Ganesh Sharma and Subhrakanti Dey. On analog distributed approximate Newton with determinantal averaging. In *2022 IEEE 33rd Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1–7. IEEE, 2022b.
- Nicolò Dal Fabbro, Subhrakanti Dey, Michele Rossi, and Luca Schenato. Shed: A newton-type algorithm for federated learning based on incremental hessian eigenvector sharing. *Automatica*, 160:111460, 2024.
- Mher Safaryan, Rustem Islamov, Xun Qian, and Peter Richtárik. FedNL: Making newton-type methods applicable to federated learning. *arXiv preprint arXiv:2106.02969*, 2021.
- David Kempe, Alin Dobra, and Johannes Gehrke. Gossip-based computation of aggregate information. In *44th Annual IEEE Symposium on Foundations of Computer Science, 2003. Proceedings.*, pages 482–491. IEEE, 2003.
- Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Gossip algorithms: Design, analysis and applications. In *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies.*, volume 3, pages 1653–1664. IEEE, 2005.
- Lin Xiao and Stephen Boyd. Fast linear iterations for distributed averaging. *Systems & Control Letters*, 53(1):65–78, 2004.
- Ali Jadbabaie, Jie Lin, and A Stephen Morse. Coordination of groups of mobile autonomous agents using nearest neighbor rules. *IEEE Transactions on automatic control*, 48(6):988–1001, 2003.
- Shreyas Sundaram and Christoforos N Hadjicostis. Finite-time distributed consensus in graphs with time-invariant topologies. In *2007 American Control Conference*, pages 711–716. IEEE, 2007.
- Guodong Shi, Bo Li, Mikael Johansson, and Karl Henrik Johansson. Finite-time convergent gossiping. *IEEE/ACM Transactions on Networking*, 24(5):2782–2794, 2015b.
- Junya Chen, Sijia Wang, Lawrence Carin, and Chenyang Tao. Finite-time consensus learning for decentralized optimization with nonlinear gossiping. *arXiv preprint arXiv:2111.02949*, 2021b.

- Edward Duc Hien Nguyen, Xin Jiang, Bicheng Ying, and César A Uribe. On graphs with finite-time consensus and their use in gradient tracking. *arXiv preprint arXiv:2311.01317*, 2023.
- Bicheng Ying, Kun Yuan, Yiming Chen, Hanbin Hu, Pan Pan, and Wotao Yin. Exponential graph is provably efficient for decentralized deep training. *Advances in Neural Information Processing Systems*, 34:13975–13987, 2021.
- Themistoklis Charalambous, Ye Yuan, Tao Yang, Wei Pan, Christoforos N Hadjicostis, and Mikael Johansson. Distributed finite-time average consensus in digraphs in the presence of time delays. *IEEE Transactions on Control of Network Systems*, 2(4):370–381, 2015.
- Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3), 2016.
- Angelia Nedic, Alex Olshevsky, and Wei Shi. Achieving geometric convergence for distributed optimization over time-varying graphs. *SIAM Journal on Optimization*, 27, 2017.
- Tsung-Hui Chang, Mingyi Hong, and Xiangfeng Wang. Multi-agent distributed optimization via inexact consensus ADMM. *IEEE Transactions on Signal Processing*, 63(2):482–497, 2014.
- Nicoletta Bof, Ruggero Carli, Giuseppe Notarstefano, Luca Schenato, and Damiano Varagnolo. Multiagent Newton–Raphson optimization over lossy networks. *IEEE Transactions on Automatic Control*, 64(7):2983–2990, 2018a.
- Damiano Varagnolo, Filippo Zanella, Angelo Cenedese, Gianluigi Pillonetto, and Luca Schenato. Newton-Raphson consensus for distributed convex optimization. *IEEE Transactions on Automatic Control*, 61(4):994–1009, 2015.
- Dragana Bajovic, Dusan Jakovetic, Natasa Krejic, and Natasa Krklec Jerinkic. Newton-like method with diagonal correction for distributed optimization. *SIAM Journal on Optimization*, 27(2), 2017.
- Boyue Li, Shicong Cen, Yuxin Chen, and Yuejie Chi. Communication-efficient distributed optimization in networks with gradient tracking and variance reduction. *The Journal of Machine Learning Research*, 21(1), 2020e.
- Lin Xiao, Stephen Boyd, and Seung-Jean Kim. Distributed average consensus with least-mean-square deviation. *Journal of parallel and distributed computing*, 67(1):33–46, 2007.

- Jorge Nocedal and Stephen J Wright. *Numerical optimization*. Springer, 1999.
- Atilim Gunes Baydin, Barak A. Pearlmutter, Alexey Andreyevich Radul, and Jeffrey Mark Siskind. Automatic differentiation in machine learning: a survey. *Journal of Machine Learning Research*, 18(153):1–43, 2018. URL <http://jmlr.org/papers/v18/17-468.html>.
- Zebang Shen, Alejandro Ribeiro, Hamed Hassani, Hui Qian, and Chao Mi. Hessian aided policy gradient. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 5729–5738. PMLR, 09–15 Jun 2019. URL <https://proceedings.mlr.press/v97/shen19d.html>.
- Aryan Mokhtari, Wei Shi, Qing Ling, and Alejandro Ribeiro. A decentralized second-order method with exact linear convergence rate for consensus optimization. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):507–522, 2016b. doi: 10.1109/TSIPN.2016.2613678.
- Alessio Maritan and Luca Schenato. ZO-JADE: Zeroth-order curvature-aware distributed multi-agent convex optimization. *IEEE Control Systems Letters*, 2023. doi: 10.1109/LCSYS.2023.3281745.
- Nicoletta Bof, Ruggero Carli, and Luca Schenato. Lyapunov theory for discrete time systems. *arXiv preprint arXiv:1809.05289*, 2018b.
- Yann LeCun, Corinna Cortes, and CJ Burges. MNIST handwritten digit database. *ATT Labs*, 2, 2010.
- Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.
- Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.