

AMERICAN UNIVERSITY OF BEIRUT

LOW COMPLEXITY DATA PROCESSING  
AND LEARNING OVER SINGLE-AGENTS  
AND DISTRIBUTED NETWORKS

by

MOHAMMAD HUSSEIN NASSRALLA

A dissertation  
submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy  
to the Department of Electrical and Computer Engineering  
of the Maroun Semaan Faculty of Engineering and Architecture  
at the American University of Beirut

Beirut, Lebanon  
November 2021

# AMERICAN UNIVERSITY OF BEIRUT

## Low Complexity Data Processing and Learning over Single-Agents and Distributed Networks

by  
Mohammad Hussein Nassralla

Approved by:

---

Dr. Zaher Dawy, Professor  
Electrical and Computer Engineering

Advisor



---

Dr. Ibrahim Abou Faycal, Professor  
Electrical and Computer Engineering

Chair of Committee



---

Dr. Hazem Hajj, Associate Professor  
Electrical and Computer Engineering

Member of Committee



---

Dr. Amr Mohamed, Professor  
College of Engineering, Qatar University

Member of Committee



Dr. Mehdi Bennis, Professor



Member of Committee

Electrical and Computer Engineering, Oulu University

---

Dr. Tareq Al-Naffouri, Professor

Member of Committee

Electrical and Computer Engineering, KAUST University



Date of thesis defense: Friday, November 5, 2021

# AMERICAN UNIVERSITY OF BEIRUT

## THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Nassralla,  
Last

Mohammad Hussein, Fawzi  
First Middle

Master's Thesis

Master's Project

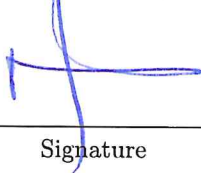
Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One** \_\_\_ year from the date of submission of my thesis, dissertation or project.

**Two**  years from the date of submission of my thesis, dissertation or project.

**Three** \_\_\_ years from the date of submission of my thesis, dissertation or project.

  
Signature

16.05.2022  
Date

This form is signed when submitting the thesis, dissertation, or project to the University Libraries

# Acknowledgements

First of all, I would like to thank my PhD advisor, Prof. Zaher Dawy, for his continuous guidance and support throughout my PhD studies. In particular, I would like to thank him for believing in my potential and agreeing to become my PhD advisor, for encouraging and enabling me to participate in scientific conferences around the world, for initiating fruitful collaborations with external partners, and for always providing meaningful ideas for future research directions.

I would also like to thank Prof. Ibrahim Abou-Faycal for acting as the chair of my PhD committee, and Prof. Hazem Hajj for serving as a committee member. Special thanks also go to Prof. Mohammed Amr (College of Engineering, Qatar University), Prof. Mehdi Bennis (Department of Electrical and Computer Engineering, Oulu University), and Prof. Tarek AlNafouri (Department of Electrical and Computer Engineering, KAUST University) for serving as external committee members.

Many thanks go to the administrative team at the EECE department of AUB for smoothly taking care of all administrative issues.

I would also like to highlight that this work was made possible by NPRP grant # NPRP12S-0305-190231 from the Qatar National Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors.

Finally, I would like to thank my parents, my wife Jinan and my daughter Fatima for their love, for believing in me and supporting my decisions, and for always being proud of my achievements!

# An Abstract of the Dissertation of

Mohammad Hussein Nassralla for Doctor of Philosophy  
Major: Electrical and Computer Engineering

Title: Low Complexity Data Processing and Learning over Single-Agents and  
Distributed Networks

An enormous amount of data is generated every second across a wide range of sectors around the globe. Learning from the available data for given use cases requires full data access, which can be challenging due to privacy considerations in addition to communications, storage and computational constraints. For these reasons, it is highly attractive to enable devices to process and learn from data locally to train a global model while exchanging only selected parameters with other devices over the network, and without the need to share and store the complete data set. This reduces the risk of privacy leakage and the communication load over the network. However, it transfers the computational burden to the end devices or local agents, which normally have energy and processing speed limitations. This PhD thesis deals with designing low complexity and effective algorithms for data processing, optimization, and learning over both single agents and networks. The thesis work is divided into three main parts. In the first part, we propose two methods for learning over networks; the first one addresses the problem of learning over heterogeneous networks and, in particular, how collaboration among heterogeneous agents can improve the overall learning performance, and the second one presents a novel low complexity approximation method for the gradient descent algorithm. Theoretical proofs of the convergence, complexity analysis, and performance results are provided and analyzed to demonstrate effectiveness and generate insights. In the second part, we present an efficient two-step design methodology for low complexity finite-impulse response (FIR)

filters. In the first step, the filter design is formulated as an optimization problem to find the minimum number of coefficients. In the second step, a mapping approach is proposed to compensate for the mean square error (MSE) that results from the approximation in the first step. Simulation results show that the designed filter has flat response in the passband, a narrow transition band, and high attenuation in the stopband. The proposed design leads to a better performance-computational complexity tradeoff compared to other state-of-the-art digital filter design methods. Finally, in the third part, a dynamic compression algorithm for EEG biomedical data is proposed. The algorithm applies a sequence of compression/decompression operations in order to find an optimized lossless/lossy compression combination that provides high compression ratio while preserving the signal integrity. Performance evaluation results on real datasets demonstrate an effective compression performance while maintaining a distortion level below a target threshold and low computational overhead.

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Background</b>	<b>3</b>
2.1 Learning over Networks: Network Topology . . . . .	3
2.2 Learning over Networks: Distributed Learning Strategies . . . . .	4
2.3 Learning over Networks: Approximation of Gradient Descent . . . . .	7
2.3.1 Mini-Batch and Stochastic Gradient Descent Algorithms . . . . .	8
<b>3 Optimization and Learning over Distributed Agents with Heterogeneous Computational Capabilities</b>	<b>10</b>
3.1 Motivation . . . . .	11
3.2 Problem Formulation . . . . .	12
3.2.1 Distributed Agents under Newton’s Method . . . . .	15
3.2.2 Distributed Agents under Quasi-Newton Method . . . . .	16
3.2.3 Distributed Agents under Gradient Descent Method . . . . .	16
3.2.4 Distributed Agents under Stochastic Gradient Descent Method . . . . .	17
3.2.5 Unifying Bounds of $Y_{k,i-1}^\delta$ . . . . .	17
3.2.6 Analysis of the Matrix $\mathcal{B}_{i-1}$ . . . . .	19
3.3 Stability Analysis . . . . .	21
3.4 Simulation Results and Performance Analysis . . . . .	22
3.4.1 Impact on Agents with High Computational Capabilities . . . . .	25
3.4.2 Impact on Agents with Low Computational Capabilities . . . . .	26
3.5 Conclusion . . . . .	29
<b>4 A Low Complexity Approximation of Gradient Descent for Learning over Single and Multi-Agent Systems</b>	<b>30</b>
4.1 Background and Literature Review . . . . .	30
4.2 Proposed Approach for Approximating the Gradient Descent Algorithm . . . . .	31



4.3	Convergence Analysis . . . . .	33
4.4	Proposed Method under Distributed Learning . . . . .	34
4.5	Simulation Results and Performance Analysis . . . . .	35
4.6	Conclusion . . . . .	37
<b>5</b>	<b>A Clustering-Based Approach for Designing Low Complexity FIR Filters</b>	<b>40</b>
5.1	Background and Literature Review . . . . .	40
5.2	Proposed Approach . . . . .	41
5.2.1	Even Symmetry of Minimizer $\{h_c^*(0), \dots, h_c^*(N - 1)\}$ . . . . .	42
5.2.2	K-means Based Clustering Solution . . . . .	44
5.2.3	Generalized Clustered FIR Filter Approximation . . . . .	44
5.3	Simulation Results and Performance Analysis . . . . .	46
5.4	Conclusion . . . . .	48
<b>6</b>	<b>Dynamic Compression for Big Biomedical Data</b>	<b>51</b>
6.1	Background and Literature Review . . . . .	51
6.2	Proposed EEG Compression Algorithm . . . . .	53
6.2.1	Training Phase . . . . .	54
6.2.2	Dynamic Parameter Adjustment Phase . . . . .	55
6.2.3	Complexity Analysis and Practical Considerations . . . . .	56
6.3	Performance Evaluation . . . . .	57
6.3.1	EEG Compression Results . . . . .	57
6.3.2	Faithfulness in Signal Reconstruction . . . . .	60
6.3.3	Complexity . . . . .	61
6.4	Conclusion . . . . .	62
<b>7</b>	<b>Future Research Directions</b>	<b>63</b>
<b>8</b>	<b>Thesis Publications</b>	<b>65</b>
	<b>Appendices</b>	<b>66</b>
	<b>Appendix A</b>	<b>67</b>
A.1	Network Mean-Square-Error Stability . . . . .	67
A.2	Network Fourth-Order Moment Stability . . . . .	69
A.3	Network Mean-Error Stability . . . . .	70
A.4	Derived Results from Lipschitz Condition . . . . .	72

# List of Figures

2.1	Network of $N$ nodes that are connected by weighted edges. Note that, we are representing the edges among agents by two separate directed arrows with weights $a_{lk}$ , $a_{kl}$ . Later on, the two arrows will be represented by single bi-directional edge. . . . .	4
3.1	A strongly connected network with $N = 10$ agents who own heterogeneous computational capabilities. Subset of the agents can run the Newton's methods, or the GD, and the remaining ones can only implement the SGD. . . . .	24
3.2	Impact of heterogeneous distributed optimization algorithms on the capable agents. . . . .	26
3.3	Evolution of the mean-square deviation learning curves over network with agents of homogeneous and heterogeneous computational capabilities under the diffusion learning strategy. Agents with limited computational capabilities employ the same step-size of $\mu = 10^{-3}$ . . . . .	28
3.4	Evolution of the empirical risk learning curves over network with agents of homogeneous and heterogeneous computational capabilities under the diffusion learning strategy. Agents with limited computational capabilities employ the same step-size of $\mu = 10^{-3}$ . . . . .	29
4.1	Comparison among the mean-square-deviation $MSD_r$ curves over single agent (agent $r=1$ in Figure 4.3) of the GD, mini-batch ( $B = 20$ ) and the proposed method ( $K = 10, 30, 60$ ). . . . .	37
4.2	Comparison among the excess risk $ER_r$ curves over single agent (agent $r=1$ in Figure 4.3) of the GD, mini-batch ( $B = 20$ ) and the proposed method ( $K = 10, 30, 60$ ). . . . .	38
4.3	A strongly connected network with $N = 10$ agents. . . . .	38
4.4	Evolution of the mean-square deviation learning curves over network with $R = 10$ agents under the diffusion learning strategy. . . . .	39
4.5	Evolution of the excess risks learning curves over network with $R = 10$ agents under the diffusion learning strategy. . . . .	39

5.1	Comparison of the frequency response of LPF ( $w_p = 0.25$ , $w_s = 0.3$ , $N = 72$ ) and HPF ( $w_p = 0.3$ , $w_s = 0.25$ , $N = 72$ ) using Hamming window, Parks-McClellan and the proposed method with $C = 20$ . . . . .	49
5.2	(a) Effect of the number of clusters on the passband and stopband gains. (b) Comparison between the proposed method and the sparse filter method of [1] with $N = 62$ , $\omega_p = 0.5$ and $\omega_s = 0.55$ . . . . .	50
6.1	Compression routine used in the training and dynamic adjustment phases. . . . .	54
6.2	Dynamic compression approach. . . . .	55
6.3	Comparison of the proposed algorithm, SPIHT and JPEG2000 with a TPRD of 10%. . . . .	59
6.4	Original and reconstructed EEG segment. . . . .	60
6.5	Fourier transform coefficients (a) and linear predictive coding (LPC) coefficients of the original and reconstructed signals. . . . .	61
6.6	Running time of proposed algorithm for different EEG segment sizes. . . . .	62

# List of Tables

5.1	Performance of PM, sparse, proposed and ideal filters in the stop-band and passband. LPF ( $\omega_p = 0.5, \omega_s = 0.55$ ), HPF ( $\omega_p = 0.55, \omega_s = 0.5$ ), N =62 coefficients, C=20 clusters (proposed) and Z=42 zeros (sparse [1]). . . . .	47
6.1	Sample testing outcome with a TPRD of 10%. . . . .	57

# Chapter 1

## Introduction

Big data is increasing daily from mobile devices, call centers, web servers, social networks, sensors, etc. Learning from distributed information and building intelligent models require full access to data. It is imperative to centralize the data at fusion centers for building smart applications and improving learning models; however, the privacy of data usually complicates or prevents its sharing. In addition, the continuous communication of big data over networks is usually limited by the link resources. On top of that, collecting big data at fusion centers ends up in storage and computational challenges. For these reasons, many techniques were proposed in the literature to develop processing and learning algorithms that work in a distributed manner. In distributed learning, the data is processed locally to train a global model without the need to share the raw information; however, few parameters are sent over the network. Local devices or agents use their own data and some shared parameters over the network to update the learning algorithm. Decentralized learning reduces the risk of privacy leakage and heavy communication but it also puts a computational burden on distributed agents which might have energy or computational capabilities restrictions. In addition, decentralized learning is susceptible to noisy reception of the shared parameters over the network, reception of parameters asynchronously, malicious attacks, etc., which might affect the accuracy of the learning model.

Learning and estimation over networks usually consist of computation, communication, decision making, etc. Computation is usually accompanied with energy consumption, while communication is usually limited by the privacy of information and the link resources (bandwidth, speed, costs, etc); in addition, decision making is usually concerned with high accuracy and good performance. This thesis deals with designing low complexity algorithms for estimation, processing, adaptation and learning over single agents and networks. Our target is to develop algorithms that have low computational complexity, not greedy for link resources and have good learning accuracy. Our thesis is mainly divided into three objectives.

The first objective consists of two parts. The first one is about learning over

heterogeneous networks, while the second is about an approximation of the gradient descent method for learning over single and multi-agent systems. In learning over heterogeneous networks, we study distributed optimization for learning problems over agents with different computational capabilities. The heterogeneity of computational capabilities implies that a subset of the agents might be able to run computationally-intensive learning algorithms like Newton’s method or full gradient descent, while the remaining ones can only run lower-complexity algorithms like stochastic gradient descent. This leads to opportunities for designing hybrid distributed optimization algorithms that rely on cooperation among the network agents in order to enhance the overall performance, improve the rate of convergence, and reduce the communication overhead. We provide a theoretical proof that hybrid learning among heterogeneous agents attains a stable solution. For small step-sizes  $\mu$ , the proposed approach leads to a small estimation error in the order of  $O(\mu)$ . Results are presented and analyzed for case study scenarios to demonstrate the effectiveness of the proposed approach.

In the second part of the first objective, we propose an approximation method for the gradient descent algorithm that is usually used over single and multi-agent systems. The proposed method computes a deterministic summary of the training data set, and then computes the learning direction per iteration with respect to the summary. We provide a convergence analysis for the proposed method and show that the thoroughness of the summary can be tweaked to optimize the trade-off between computational complexity and convergence rate. Performance results are illustrated numerically for single and multi-agent systems.

In the second objective of the thesis, we present an efficient two-step method to design low complexity finite-impulse response (FIR) filters. In the first step, the filter design is formulated as an optimization problem to find the minimum number of coefficients. In the second step, a mapping approach is proposed to compensate for the mean square error (MSE) that results from the first step. The proposed method demonstrates its effectiveness in terms of computational complexity without compromising the filtering performance.

Finally, in the third objective, we propose a dynamic and effective compression approach for large data sets of brain signals that are acquired by the electroencephalogram (EEG) technique. The proposed compression algorithm relies on a sequence of compression and decompression phases to optimize the compression rate while maintaining a distortion level below a target threshold. The proposed approach shows high compression ratios while it maintains the signals quality.

The remaining parts of this thesis are organized as follows. Chapter 2 provides background information about learning over single agents and networks. Chapter 3 presents the proposed work about learning over heterogeneous networks, while Chapter 4 presents our work for approximating the gradient descent algorithm. Chapter 5 presents the proposed novel approach for designing low complexity FIR filters, while Chapter 6 describes our work on biomedical data compression. Finally, Chapter 7 presents future research directions.

# Chapter 2

## Background

### 2.1 Learning over Networks: Network Topology

Optimization and learning over single and multi-agent systems usually entails a network of  $N$  interacting agents, labeled as  $k = 1, 2, \dots, N$ . The network is usually characterized by a graph involving  $N$  vertices (nodes, devices, or agents) and a set of edges connecting the vertices to each other as shown in Figure 2.1. We assume the graph to be undirected so that if agent or node  $k$  is a neighbor of node  $l$ , then node  $l$  is also a neighbor of agent  $k$  [2]. Neighbor nodes can share data both ways over the edge joining them. Every edge connecting two agents will be assigned a weight  $a_{lk}$  to form up a *weighted* graph [2]. The subscripts  $l$  and  $k$  in  $a_{lk}$  represent the source and the sink respectively, where  $a_{lk}$  is used by node  $k$  to scale the information it receives from  $l$ .

Network topology plays an important role in learning over networks, and there are many network structures in the literature such as strongly connected networks, connected networks, weakly connected networks, etc. In this work, we focus on distributed learning over strongly connected networks. Strongly-connected network means that there always exists at least one path connecting any two agents, and that there is at least one agent with self connecting loop. This indicates that there is at least one agent in the network that relies on its own data and will allocate a positive weight to it. The strong connectivity of a network turns out into a useful property that can be understood well if we represent the weights of a network by  $N \times N$  adjacency matrix  $A$ , where the elements on the  $k$ -th column of  $A$  include the weights used by agent  $k$  to scale information incoming from its neighbors  $l \in \mathcal{N}_k$ ; we set  $a_{lk} = 0$  if  $l \notin \mathcal{N}_k$ . So, the row and column indices in  $(l, k)$  denote the source and destination agents respectively. So the topology of the network is summarized by the matrix  $A$ , which is a very important matrix in learning over networks, and it plays an important role in the stability and convergence analysis. It has a useful property that it is *primitive*. The primitiveness of  $A$  allows the use of the *Perron – Frobenius* theorem in

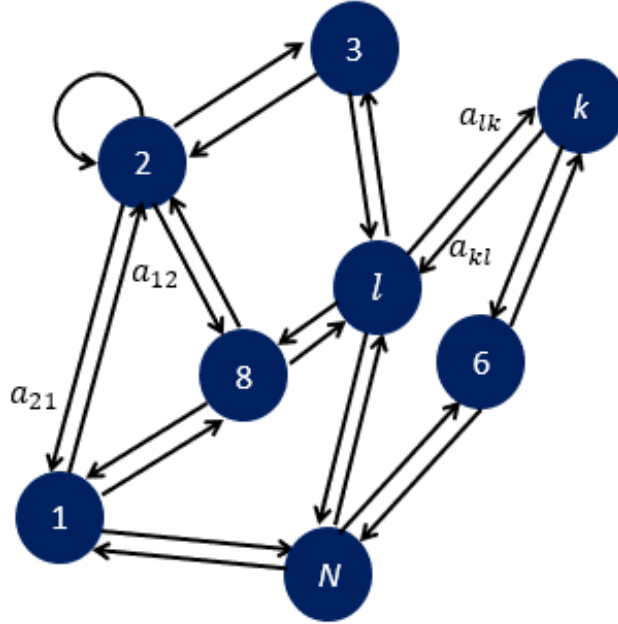


Figure 2.1: Network of  $N$  nodes that are connected by weighted edges. Note that, we are representing the edges among agents by two separate directed arrows with weights  $a_{lk}$ ,  $a_{kl}$ . Later on, the two arrows will be represented by single bi-directional edge.

matrix theory to prove the convergence of several algorithms over the network later on. The *Perron – Frobenius* theorem describes the eigen-structure of  $A$  as follows

- Matrix  $A$  has a single eigenvalue at one, and all other eigenvalues of  $A$  are strictly inside the unit circle (the spectral radius of  $A$  is  $\rho(A)$  and it is equal to one).
- With proper sign scaling, all entries of the right-eigen vector of  $A$  corresponding to the single eigenvalue at one are positive.

## 2.2 Learning over Networks: Distributed Learning Strategies

Every agent  $k$  over the network holds its own data and works to minimize its cost function  $J_k(\omega)$  in collaboration with other dispersed agents in its neighborhood  $\mathcal{N}_k$ . Every agent attains its minimizer locally where the local minimizer over the network  $\omega^o$  corresponds to the minimizer of the weighted aggregate cost  $J^{glob}(\omega)$ ,



namely

$$\omega^o = \underset{\omega}{\operatorname{argmin}} J^{glob}(\omega) = \underset{\omega}{\operatorname{argmin}} \sum_{k=1}^N \alpha_k J_k(\omega) \quad (2.1)$$

where  $J^{glob}(\omega)$  is given by

$$J^{glob}(\omega) = \sum_{k=1}^N \alpha_k J_k(\omega) \quad (2.2)$$

where  $\alpha_k$  is some positive scalar,  $\alpha_k \geq 0$ .

The following is an assumption about the individual and the aggregate cost functions  $J_k(\omega)$ , and  $J^{glob}(\omega)$ .

**Assumption 1 (Conditions on Aggregate and Individual Costs)**

Assume that the individual costs,  $J_k(\omega)$ , are each twice-differentiable and convex, with at least one of them being  $\nu$ -strongly convex. It follows that  $J^{glob}(\omega)$  is strongly convex and attains a global minimizer  $\omega^o$ , and satisfies

$$\nabla J^{glob}(\omega^o) = \sum_{k=1}^N \alpha_k \nabla J_k(\omega^o) = 0 \quad (2.3)$$

In addition, we assume that the gradient  $\nabla J_k(\omega)$  is  $\delta_c$ -Lipschitz. It follows that the Hessian of the individual cost function  $\nabla^2 J_k(\omega)$ , and the Hessian of the aggregate cost function  $\nabla^2 J^{glob}(\omega)$  satisfy the following conditions

$$\nabla^2 J_k(\omega) \leq \delta_c I_M \quad (2.4)$$

$$\nu I_M \leq \nabla^2 J^{glob}(\omega) \leq \delta_c I_M \quad (2.5)$$

$$\nabla^2 J_{k_0}(\omega) \geq \nu I_M > 0, \quad \nabla^2 J_k(\omega) \geq 0, \quad k \neq k_0 \quad (2.6)$$

for some positive parameter  $\nu \leq \delta_c$ . ■

There exists many distributed learning strategies like incremental [2], consensus [2], [3], diffusion [2], [4], [5], enlarged cooperation [2], spatio-temporal [6], etc. In these methods, every agent  $k$  works to minimize (2.1) by using two main steps; a combination step and an adaptation step. In the combination step, every agent combines linearly the shared iterates over the network from agents in its neighborhood, while in the adaptation step an iterative optimization algorithm is run by every agent. We adopt in this work the diffusion method due to its stability and superior adaptation performance. Diffusion techniques have shown better performance in adaptive situations where it is essential to track drifts in the underlying models through constant step-size adaptation [7], [8]. The two

steps of combination and adaptation under the diffusion technique are given as follows

$$\left\{ \begin{array}{l} \textbf{Combination Step:} \\ \psi_{k,i-1} = \sum_{l \in \mathcal{N}_k} a_{lk} \omega_{l,i-1} \\ \textbf{Adaptation Step:} \\ \omega_{k,i} = \psi_{k,i-1} + \mu_k d_k(\psi_{k,i-1}) \end{array} \right. \quad (2.7)$$

where  $\omega_{k,i} \in R^M$  is the global parameter, in which all agents over the network must decide upon,  $\psi_{k,i-1}$  combines linearly the previous iterates  $\omega_{l,i-1}$  received from the neighborhood of agent  $k$  and then  $\psi_{k,i-1}$  is used to update the descent direction optimization algorithm as given in the second line of (2.7),  $d_k^j(\cdot)$  is the descent direction, and  $\mu_k$  is the learning rate or the step size. The coefficient  $a_{lk}$  that appear in (2.7) represents the weight that agent  $k$  assigns to the received iterate  $\omega_{l,i-1}$  from agent  $l$ , and these weights are usually selected to satisfy the following conditions [9–16],

$$a_{lk} \geq 0, \quad \sum_{l=1}^N a_{lk} = 1, \quad \text{and } a_{lk} = 0 \text{ if } l \notin \mathcal{N}_k, \quad k = 1, 2, \dots, N \quad (2.8)$$

The entries  $\{a_{lk}\}$  can be collected into an  $N \times N$  matrix  $A_1$ , such that the  $k$ -th column of  $A_1$  consists of  $\{a_{lk}, l = 1, 2, \dots, N\}$ .

The descent direction  $d_k(\cdot)$  may take different forms depending on the adopted iterative optimization algorithm. There exists many descent directions in literature [17], [18]. Below are the most well known descent directions in the numerical optimization field

$$d_{k,i-1} = \begin{cases} -[\nabla^2 J_k(\psi_{k,i-1})]^{-1} \nabla J_k(\psi_{k,i-1}) & \text{for Newton's method,} \\ -B_k(\psi_{k,i-1}) \nabla J_k(\psi_{k,i-1}) & \text{for Quasi-Newton method,} \\ -\nabla J_k(\psi_{k,i-1}) & \text{for gradient descent method,} \\ -\widehat{\nabla J_k}(\psi_{k,i-1}) & \text{for stochastic gradient descent method.} \end{cases} \quad (2.9)$$

Newton's method is known to be the fastest to converge iterative optimization algorithm, however it requires the computation of the inverse of the Hessian matrix at every iteration which represents a burden for most of the distributed agents. The Quasi-Newton method is an approximation technique for Newton's method and it can compute efficiently the inverse of the Hessian matrix by relying only on the gradient of the cost function [17], [18]. There are many different versions of Quasi-Newton methods, but they are all based on approximating the inverse of the Hessian  $[\nabla^2 J(\psi_{k,i-1})]^{-1}$  by another matrix  $B(\psi_{k,i-1})$  which may take different forms depending on the update formula such as the techniques of Broyden-Fletcher-Goldfarb-Shanno (BFGS), Davidon-Fletcher-Powell

(DFP), etc., [17], [18]. Even though quasi-Newton approximates well the Newton's method, but it is still computationally demanding. Gradient descent (GD) represents a good alternative to Newton's methods when it is essential to reduce the computational complexity at every iteration. However, in some applications, the optimization problems entail thousands or millions of data samples which makes the task of computing the gradient for the whole data set a challenging one. Also, in online optimization algorithms there is a tendency to reduce the computational complexity of gradient descent to gain in speed and achieve the real time requirements. Problems of these type can be solved efficiently by approximating the gradient by stochastic gradient  $\widehat{\nabla}J(\psi_{k,i-1})$  as it is usually done in mini-batch and stochastic gradient descent.

We propose in the next two chapters two different approaches to benefit from the network topology and the data structure to improve the convergence rate of learning over networks, which leads to reducing the communication overhead and decreases the computational complexity at every distributed agent. However, before continuing with the proposed work in the following chapters, we summarize the most prevalent methods for approximating the gradient as it is important for the remaining parts of the thesis.

## 2.3 Learning over Networks: Approximation of Gradient Descent

Equation (2.9) reveals that the descent directions require at least the computation of the gradient of the cost function at every iteration. The cost function  $J_k(\psi_{k,i-1})$  and its gradient are usually expressed as a summation of loss function  $Q_k(\psi_{k,i-1})$  and  $\nabla Q_k(\psi_{k,i-1})$  as follows:

$$J_k(\psi_{k,i-1}) = \frac{1}{L} \sum_{n=0}^{L-1} Q_k(\psi_{k,i-1}, \gamma_k(n), h_k(n)), \quad h_k \in R^m, \gamma_k(n) \in R. \quad (2.10)$$

$$\nabla J_k(\psi_{k,i-1}) = \frac{1}{L} \sum_{n=0}^{L-1} \nabla Q_k(\psi_{k,i-1}, \gamma_k(n), h_k(n)), \quad h_k \in R^m, \gamma_k(n) \in R, \quad (2.11)$$

where  $h_k(n)$ ,  $\gamma_k(n)$ , and  $L$  represent the data set, the label/class of the data, and the size of the data set, respectively.

The computation of the gradient becomes intractable and a challenging task, when it comes to big data set, online learning, or agents with low computational complexity. So it is proposed in the literature to approximate the computing of the gradient at every iteration, and there exists many low complexity algorithms,

however, this comes at the expense of slow convergence rates, which ends up in running the network for a long time and thus increases the communication overhead. We will outline in the following section the different algorithms that are widely used to approximate the gradient to deal with online learning and big-data scenarios. Then in the next chapters, we will present our work to improve the convergence speed of learning over single and multi-agent systems.

### 2.3.1 Mini-Batch and Stochastic Gradient Descent Algorithms

Gradient descent or batch learning uses the whole data set at every iteration to learn the parameters that minimize the desired cost function as shown below [17], [2]

$$\begin{cases} \psi_{k,i-1} = \sum_{l \in \mathcal{N}_k} a_{lk} \omega_{l,i-1} \\ \omega_{k,i} = \psi_{k,i-1} - \mu \left[ \frac{1}{L} \sum_{n=0}^{L-1} \nabla Q_k(\psi_{k,i-1}, \gamma_k(n), h_k(n)) \right]. \end{cases} \quad (2.12)$$

Batch learning needs  $O(n)$  computations at every iteration (since we have to calculate the  $\nabla Q(\omega)$  for all  $L$  samples of the data). When the size  $L$  of the data set is large, it is impractical to use (2.12) since it is costly to evaluate the gradient over the whole data set. Different algorithms are proposed in the literature to approximate the gradient computation at every iteration by  $\widehat{\nabla J}_k(\cdot)$ . The approximation is based on selecting a subset of the data samples at every iteration to compute the gradient of the cost function. For instance, mini-batch gradient descent approximates (2.12) by uniformly selecting at each iteration a subset  $B$  of the  $L$  training data samples  $\{\gamma_k(n), h_k(n)\}$  [19], [20]. The mini-batch  $GD$  is given by

$$\begin{cases} \psi_{k,i-1} = \sum_{l \in \mathcal{N}_k} a_{lk} \omega_{l,i-1} \\ \omega_{k,i} = \psi_{k,i-1} - \mu \widehat{\nabla J}_k(\psi_{k,i-1}, \gamma_k(b), h_k(b)) \\ \omega_{k,i} = \psi_{k,i-1} - \mu \left[ \frac{1}{B} \sum_{b=0}^{B-1} \nabla Q_k(\psi_{k,i-1}, \gamma_k(b), h_k(b)) \right] \end{cases} \quad (2.13)$$

where  $B$  is the mini-batch size and  $B < L$ . The  $B$  samples can be selected in various ways as follows

- Uniform sampling with replacement: In this case, the indices of the  $B$  samples are selected uniformly from the set of indices  $\{0, 1, \dots, L-1\}$  so that, for each integer  $i$  in this set,  $J(b=i) = \frac{1}{L}$ .

- Random sampling: The indices of the  $B$  samples are selected here according to the following distribution:

$$P(b = i) = p_i, \quad i \in \{0, 1, \dots, L - 1\} \quad (2.14)$$

where  $p_i$  represents the occurrence probability of the samples in the data set.

- Data streaming: The  $B$  training samples are chosen as the most recent  $B$  samples in a streaming implementation. Note that, the sampling method doesn't affect the complexity of the algorithm.

*SGD* is another algorithm to approximate the *GD* by selecting at random one pair ( $B = 1$ ) of data samples  $\{\gamma_k(n), h_k(n)\}$  at every iteration [21]. At every iteration, the data sample  $\{\gamma_k(n), h_k(n)\}$  can be chosen as presented in the aforementioned methods, or the  $N$  samples in the data set can be randomly reshuffled and then the data are selected from the reshuffled set sequentially in an increasing order.

It is useful to mention that the approximation of the gradient introduces a gradient noise, which represents the difference between the full gradient  $\nabla J_k(\cdot)$  and the approximated gradient  $\widehat{\nabla J}_k(\cdot)$  and it is defined as follows

$$s_{k,i}^\delta(\psi_{k,i-1}) = \widehat{\nabla J}_k(\psi_{k,i-1}) - \nabla J_k(\psi_{k,i-1}), \quad (2.15)$$

The mini-batch and stochastic gradient descent represent an appealing solution to approximate the *GD* algorithm, however, they need a lot of iterations to converge. Many techniques are proposed in the literature to improve the speed and performance of mini-batch and SGD algorithms based on either adapting the learning rate, adapting the gradient or adapting both the learning rate and the gradient. Among these enhanced techniques are the momentum [22], Nesterov accelerated gradient (NAG) [23], Adagrad [24], Adadelata [25], Adam [26], Nadam [27] and AMSGrad [28].

## Chapter 3

# Optimization and Learning over Distributed Agents with Heterogeneous Computational Capabilities

Distributed optimization is a key enabler for collaborative learning in intelligent networks in which the edge devices only have access to their local data streams. It provides a scalable solution to the distributed inference problem in scenarios where the communication between nodes is costly and data centralization raises privacy and computational concerns [29–35]. Decentralized learning can overcome the challenges of privacy, communication overhead, and the computational burden at centralized agents by processing the information locally at distributed agents. In this context, each agent senses data, processes it and shares some learning parameters over the network. Every agent runs an iterative optimization algorithm to minimize a certain cost function and sends its local updates to other agents over the network.

Learning over networks induces many degrees of freedom and there are many factors that can improve or degrade the distributed learning performance. For instance, the following design options have an imperative impact on the performance of distributed learning algorithms: the optimization algorithm that is being run by every agent, the distributed learning strategy, the selection of the weights over the edges of the network, the synchronous/asynchronous reception of shared parameters over the network [36–38], the topology of the network [39], [2] (strongly connected, weakly connected, etc.), the link failures [40], the exchange of noisy information [41–43], the malicious behaviour [44], etc. The weights over the edges of the network play an important role in improving the minimization of the cost function of every agent and they are chosen according to different rules like the metropolis [9], [10], Hasting [11], [12] relative degree [13], Laplacian [14–16], etc.

There exist many open problems that have to be addressed in learning over networks. In this part of the thesis, we propose to study two main points; learning over heterogeneous networks, and distributed learning under summary of data. In the first point, we mean by the heterogeneous learning when the distributed agents have different computational capabilities. The heterogeneity of computational capabilities implies that a subset of the agents may run computationally-intensive learning algorithms like Newton’s method or full gradient descent, while the other agents can only run lower-complexity algorithms like stochastic gradient descent. This leads to opportunities for designing hybrid distributed optimization algorithms that rely on cooperation among the network agents in order to enhance overall performance, improve the rate of convergence, and reduce the communication overhead. On the other side, we mean by the distributed learning under summarized data, is to approximate the data intelligently such that it can help in reducing the computational complexity of the gradient descent algorithm locally for every agent. The remaining parts of this chapter present our work in learning over heterogeneous networks, while the next chapter presents our work about the approximation of the gradient descent algorithm and the learning under summarized data samples.

### 3.1 Motivation

Most prior literature focuses on distributed optimization in homogeneous networks where all agents have similar computational capabilities and apply the same learning algorithm. Having similar computational capabilities means that the distributed agents should run the same iterative optimization algorithm. For instance, the methods of incremental [2], consensus [2], [3], diffusion [2], [4], [5], enlarged cooperation [2], spatio-temporal [6], etc., are among the distributed strategies that are proposed in the literature where all agents are assumed to have homogeneous computational capabilities. However, real network deployments have much richer structure and may comprise agents with various energy constraints, hardware complexities and different computational power [45], [46], [47], [48]. In this work, we study learning over networks when the distributed agents have different or heterogeneous computational capabilities and can run different iterative optimization algorithms while cooperating among each other. For instance, a subset of the capable agents can run the Newton’s methods, full gradient descent, etc., while the other subset might only be able to run stochastic gradient descent (SGD) due to their limited computational capabilities.

There exist many applications for optimizing learning over multiple agents with heterogeneous computational capabilities. For instance, in wireless cellular networks with cooperative and distributed learning, some mobile phones may have more computational power than others, and base stations may serve as edge

computing nodes with much higher capabilities than mobile phones [33], [49], [50], [51]. In wireless sensor networks (WSN), sensor devices normally have varying levels of computational, energy, and storage resources, which restricts their ability to run different classes of distributed learning algorithms [52], [53], [54], [55], [56]. Mixed models of hybrid optimization algorithms can also find application use cases in distributed power systems, computer networks, industrial control systems, etc.

The main contribution in this part of the thesis is in proposing a hybrid distributed optimization approach for learning over networks with heterogeneous agents and in proving theoretically its stability. The theoretical and simulation results show that the mean-error process, the mean-square error, and the fourth order moment of the error process  $\|\omega^o - \omega_{k,i}\|$  converge to a stationary point and attain a stable solution. This work shows that the cooperation among agents that can run fast algorithms like Newton’s methods and slow ones like SGD can improve the rate of convergence and performance of incapable agents. Improving the rate of convergence and performance for agents with limited computational capabilities leads to reducing the local power consumption, decreasing the communication overhead over the network and saving financial costs resulting from running the network for a long time.

## 3.2 Problem Formulation

In this work, we assume a network of  $N$  interacting agents with different computational capabilities, which allows the distributed agents to run heterogeneous optimization algorithms. The individual cost functions are denoted by  $J_k(\omega)$ . Different algorithms might be run over the network and we define a parameter  $\delta$  to distinguish among them as follows

$$\delta = \begin{cases} NM & \text{for agents with high computational capabilities} \\ & \text{and run the Newton’s method,} \\ QN & \text{for agents with high computational capabilities} \\ & \text{and run the Quasi-Newton method,} \\ GD & \text{for agents with high computational capabilities} \\ & \text{and run the gradient descent method,} \\ SGD & \text{for agents with low computational capabilities} \\ & \text{and run the stochastic gradient descent method.} \end{cases} \quad (3.1)$$

Note that, we only focus on Newton’s, Quasi-Newton, gradient descent, and SGD methods since these are the widely used techniques in iterative optimization algorithms.

As stated in Section 2.2, we adopt the diffusion as a distributed learning



strategy for its stability and superior adaptation performance. We repeat below the equation of the diffusion method that was given in (2.7), but we modify it to take into consideration the heterogeneous algorithms that are being run by dispersed agents. We rewrite (2.7)

$$\begin{cases} \psi_{k,i-1} = \sum_{l \in \mathcal{N}_k} a_{lk} \omega_{l,i-1}, \\ \omega_{k,i} = \psi_{k,i-1} + \mu_k d_k^\delta(\psi_{k,i-1}) \end{cases} \quad (3.2)$$

where  $d_k^\delta(\cdot)$  is the descent direction. We suggest the following definition for  $d_k^\delta(\cdot)$  to accommodate all possibilities of the descent directions that might be adopted by the distributed agents.

$$d_k^\delta(\psi_{k,i-1}) = -Y_{k,i-1}^\delta \{ \nabla J_k(\psi_{k,i-1}) + s_{k,i}^\delta(\psi_{k,i-1}) \} \quad (3.3)$$

The matrix  $Y_{k,i-1}^\delta \in R^{M \times M}$  and its value depends on the adopted iterative optimization algorithm as shown below

$$Y_{k,i-1}^\delta = \begin{cases} [\nabla^2 J_{k,i-1}(\psi_{k,i-1})]^{-1} & \text{when } \delta = NM, \\ B_{k,i-1}(\psi_{k,i-1}) & \text{when } \delta = QN, \\ I_M & \text{when } \delta = GD, \\ I_M & \text{when } \delta = SGD. \end{cases} \quad (3.4)$$

Computing the matrix  $Y_{k,i-1}^\delta$  at the minimizer  $\omega^o$ , produces a matrix  $Y_{k,o}^\delta$  which is defined as follows

$$Y_{k,o}^\delta = \begin{cases} [\nabla^2 J_{k,i-1}(\omega^o)]^{-1} & \text{when } \delta = NM, \\ B_{k,i-1}(\omega^o) & \text{when } \delta = QN, \\ I_M & \text{when } \delta = GD, \\ I_M & \text{when } \delta = SGD. \end{cases} \quad (3.5)$$

Note that  $s_{k,i}^\delta(\psi_{k,i-1})$  is the gradient noise that was defined in (2.15), but we modify it here to accommodate for the different optimization algorithms that are being run over the network.

$$s_{k,i}^\delta(\psi_{k,i-1}) = \begin{cases} 0, & \text{when } \delta = NM, QN, \text{ or } GD, \\ \widehat{\nabla J_k}(\psi_{k,i-1}) - \nabla J_k(\psi_{k,i-1}), & \text{when } \delta = SGD. \end{cases} \quad (3.6)$$

Using (3.3) we can write (3.2) as follows

$$\begin{cases} \psi_{k,i-1} = \sum_{l \in \mathcal{N}_k} a_{lk} \omega_{l,i-1}, \\ \omega_{k,i} = \psi_{k,i-1} - \mu_k Y_{k,i-1}^\delta \{ \nabla J_k(\psi_{k,i-1}) + s_{k,i}^\delta(\psi_{k,i-1}) \} \end{cases} \quad (3.7)$$

Let  $\tilde{\omega}_{k,i}$ , and  $\tilde{\psi}_{k,i-1}$  be defined as follows

$$\tilde{\omega}_{k,i} = \omega^o - \omega_{k,i} \quad (3.8)$$

$$\tilde{\psi}_{k,i-1} = \omega^o - \psi_{k,i} \quad (3.9)$$

where  $\omega^o$  is the optimal minimizer of the global cost function given in (2.2). By using the mean value theorem [57], it can be shown that  $\nabla J_k(\psi_{k,i-1})$  can be written as

$$\nabla J_k(\psi_{k,i-1}) = -H_{k,i-1}\tilde{\psi}_{k,i-1} - b_k \quad (3.10)$$

where  $b_k$  is the gradient of the cost function at the optimal value of the network and it is given by

$$b_k = -\nabla J_k(\omega^o) \quad (3.11)$$

$H_{k,i-1}$  is defined to be

$$H_{k,i-1} = \int_0^1 \nabla^2 J_k(\omega^o - t\tilde{\psi}_{k,i-1}) dt \quad (3.12)$$

From (2.4), (3.12) and since each individual Hessian matrix is at least non-negative definite, we get

$$H_{k,i-1} \leq \delta_c I_M \quad (3.13)$$

Moreover, we deduce from (2.6), and (3.12) that

$$H_{k_o}(\omega) \geq \nu I_M > 0, \quad H_k \geq 0, \quad k \neq k_o \quad (3.14)$$

Subtracting  $\omega^o$  from both sides of (3.7) and using (3.8), (3.9) and (3.10), then (3.7) can be rewritten as

$$\begin{cases} \tilde{\psi}_{k,i-1} = \sum_{l \in \mathcal{N}_k} a_{lk} \tilde{\omega}_{l,i-1}, \\ \tilde{\omega}_{k,i} = \tilde{\psi}_{k,i-1} - \mu_k Y_{k,i-1}^\delta H_{k,i-1} \tilde{\psi}_{k,i-1} - \mu_k Y_{k,i-1}^\delta b_k + \mu_k Y_{k,i-1}^\delta s_{k,i}^\delta(\psi_{k,i-1}) \end{cases} \quad (3.15)$$

To assess the evolution of the error dynamics across the entire network, equations (3.15) of all agents can be collected and written into a network-error recursion formula as follows

$$\tilde{\omega}_i = \mathcal{B}_{i-1} \tilde{\omega}_{i-1} + \mathcal{M} s_i(\omega_{i-1}) - \mathcal{M} b \quad (3.16)$$

where

$$\tilde{\omega}_i = \text{col}\{\tilde{\omega}_{1,i}, \tilde{\omega}_{2,i}, \dots, \tilde{\omega}_{N,i}\} \quad (3.17)$$

$$\mathcal{B}_{i-1} = (\mathcal{I} - \mathcal{M} \mathcal{H}'_{i-1}) \mathcal{A}_1^T \quad (3.18)$$

$$\mathcal{I} = I_N \otimes I_M \quad (3.19)$$

$$\mathcal{A}_1 = A_1 \otimes I_M \quad (3.20)$$

where  $A_1$  is a left stochastic matrix that includes the weights  $\{a_{lk}\}$  over the edges of the network, and the term  $\otimes$  is the Kronecker product. Note that the weights  $\alpha_k$  that are used in (2.2) are chosen to be

$$\alpha_k = \mu_k p_k \quad (3.21)$$

where  $p_k$  represents the  $k$ -th entry of the Perron eigenvector of  $A_1$  ( $A_1 p = p$ ,  $1^T p = 1$ ).

$$\mathcal{M} = \text{diag}\{\mu_1 I_M, \mu_2 I_M, \dots, \mu_N I_M\} \quad (3.22)$$

$$\mathbf{s}_i(\boldsymbol{\omega}_{i-1}) = \text{col}\{Y_{k,i-1}^\delta s_{1,i}^\delta(\psi_{1,i-1}), \dots, Y_{k,i-1}^\delta s_{N,i}^\delta(\psi_{N,i-1})\} \quad (3.23)$$

$$\mathbf{b} = \text{col}\{Y_{1,o}^\delta b_1, Y_{2,o}^\delta b_2, \dots, Y_{N,o}^\delta b_N\} \quad (3.24)$$

$$\mathbf{H}'_{i-1} = \text{diag}\{\mathbf{H}_{1,i-1}^\delta, \mathbf{H}_{2,i-1}^\delta, \dots, \mathbf{H}_{N,i-1}^\delta\} \quad (3.25)$$

$$\mathbf{H}_{k,i-1}^\delta = Y_{k,i-1}^\delta H_{k,i-1} \quad (3.26)$$

According to (3.1), different forms can be taken by  $d_k^\delta(\psi_{k,i-1})$  and  $Y_{k,i-1}^\delta$ . In addition, we state the lower and upper bounds of  $Y_{k,i-1}^\delta$ , which will be useful in the remaining parts of this thesis.

### 3.2.1 Distributed Agents under Newton's Method

When a subset of the agents runs the Newton's method, then the descent direction  $d_k^{\delta=NM}(\psi_{k,i-1})$  in (3.2) becomes

$$d_k^{\delta=NM}(\psi_{k,i-1}) = -Y_{k,i-1}^{\delta=NM} \nabla J_{k,i-1}(\cdot) \quad (3.27)$$

where  $s_{k,i}^{\delta=NM}(\psi_{k,i-1})$  is equal to zero, and  $Y_{k,i-1}^{\delta=NM}$  in (3.7) is given in this case by

$$Y_{k,i-1}^{\delta=NM} = [\nabla^2 J_{k,i-1}(\psi_{k,i-1})]^{-1} \quad (3.28)$$

and  $[\nabla^2 J_{k,i-1}(\cdot)]^{-1}$  is the inverse of the Hessian matrix at every iteration. The induced 2-norm of  $Y_{k,i-1}^{\delta=NM}$  can be upper bounded as follows

$$\begin{aligned} \|Y_{k,i-1}^{\delta=NM}\| &= \|[\nabla^2 J_{k,i-1}(\psi_{k,i-1})]^{-1}\| \\ &\stackrel{a}{\leq} 2 \|[\nabla^2 J_k(\omega_k^o)]^{-1}\| = \tilde{L}_k \end{aligned} \quad (3.29)$$

where  $\tilde{L}_k$  is some positive number, and (a) follows since  $\nabla^2 J_k(\cdot)$  is assumed to be positive definite at every iteration, then  $\nabla^2 J_k(\omega_k^o)$  is non-singular and thus there is a radius  $r > 0$  such that  $\|[\nabla^2 J_k(\psi_{k,i-1})]^{-1}\| \leq 2 \|[\nabla^2 J_k(\omega_k^o)]^{-1}\|$  for all  $\omega_k$  with  $\|\psi_{k,i-1} - \omega_k^o\| \leq r$  [17]. Note that if  $[\nabla^2 J_k(\psi_{k,i-1})]^{-1}$  is not positive definite at every iteration, then it can be replaced by a related positive-definite matrix like modified Hessian matrices to guarantee that condition [17], [18]. This means that  $Y_{k,i-1}^{\delta=NM}$  can be upper bounded as follows

$$Y_{k,i-1}^{\delta=NM} \leq c_2 I_M \quad (3.30)$$

where  $c_2$  is some positive number. In addition, since  $\nabla^2 J_{k,i-1} \leq \delta_c I_M$ , then

$$Y_{k,i-1}^{\delta=NM} \geq \frac{1}{\delta_c} I_M = c_1 I_M \quad (3.31)$$

Consequently

$$c_1 I_M \leq Y_{k,i-1}^{\delta=NM} \leq c_2 I_M \quad (3.32)$$

### 3.2.2 Distributed Agents under Quasi-Newton Method

Quasi-Newton method is an approximation technique for the Newton's method and it can compute efficiently the inverse of the Hessian matrix by relying only on the gradient of the cost function [17], [18], [58]. There are many different versions of Quasi-Newton methods, but they are all based on approximating the inverse of the Hessian  $[\nabla^2 J_k(\cdot)]^{-1}$  by another matrix  $B_k$  which may take different forms depending on the update formula such as the techniques of Broyden-Fletcher-Goldfarb-Shanno (BFGS), Davidon-Fletcher-Powell (DFP), etc. So,  $Y_{k,i-1}^{\delta=QN}$  usually takes the following form under Quasi-Newton method

$$Y_{k,i-1}^{\delta=QN} = B_{k,i-1} \quad (3.33)$$

Then the descent direction  $d_k^{\delta=QN}(\psi_{k,i-1})$  in (3.2) becomes

$$d_k^{\delta=QN}(\psi_{k,i-1}) = -B_{k,i-1} \nabla J_{k,i-1}(\cdot) \quad (3.34)$$

Note that  $s_{k,i}^{\delta=QN}(\psi_{k,i-1})$  is equal to zero, and since  $B_k$  is designed to be symmetric and positive definite at every iteration, then  $Y_{k,i-1}^{\delta=QN}$  can be bounded as follows

$$c_{lq} I_M \leq Y_{k,i-1}^{\delta=QN} \leq c_{uq} I_M \quad (3.35)$$

### 3.2.3 Distributed Agents under Gradient Descent Method

When the gradient descent is being implemented by any of the distributed agents, then the  $s_{k,i}^{\delta=GD}(\psi_{k,i-1})$  is equal to zero, and the  $Y_{k,i-1}^{\delta=GD}$  in (3.7) is given in this case by

$$Y_{k,i-1}^{\delta=GD} = I_M \quad (3.36)$$

The descent direction  $d_k^{\delta=GD}(\psi_{k,i-1})$  in (3.2) becomes

$$d_k^{\delta=GD}(\psi_{k,i-1}) = -\nabla J_{k,i-1}(\psi_{k,i-1}) \quad (3.37)$$

The induced 2-norm of  $Y_{k,i-1}^{\delta=GD}$  is given by

$$\|Y_{k,i-1}^{\delta=GD}\| = 1 \quad (3.38)$$

### 3.2.4 Distributed Agents under Stochastic Gradient Descent Method

Stochastic Gradient Descent (*SGD*) is an iterative optimization algorithm that is used to approximate the gradient descent and it is considered a promising solution for large datasets and online learning [59], however it can be slow to converge [22], [24], [27], [28], [60]. The  $s_{k,i}^{\delta=SG}(\psi_{k,i-1})$  is non-zero, and  $Y_{k,i-1}^{\delta=SG}$  in (3.7) is given in this case by

$$Y_{k,i-1}^{\delta=SG}(\psi_{k,i-1}) = I_M \quad (3.39)$$

Then, the descent direction  $d_k^{\delta=SG}(\psi_{k,i-1})$  in (3.2) for agents who implement the SGD is given by

$$d_k^{\delta=SG}(\psi_{k,i-1}) = -[\nabla J_k(\psi_{k,i-1}) + s_{k,i}^{\delta=SG}(\psi_{k,i-1})] \quad (3.40)$$

The induced 2-norm of  $Y_{k,i-1}^s$  is given by

$$\|Y_{k,i-1}^{\delta=SG}\| = 1 \quad (3.41)$$

### 3.2.5 Unifying Bounds of $Y_{k,i-1}^\delta$

We capture in this section the various bounds of  $Y_{k,i-1}^\delta$  by a single unifying description under different iterative optimization strategies. We conclude from (3.32), (3.35), (3.36), and (3.39) that  $Y_{k,i-1}^\delta$  is bounded as follows

$$c'_1 I_M = \min\{1, c_1, c_{uq}\} I_M \leq Y_{k,i-1}^\delta \leq \max\{1, c_2, c_{uq}\} I_M = c'_2 I_M \quad (3.42)$$

In addition, we conclude from (3.29), (3.35), (3.38), and (3.41) that

$$\|Y_{k,i-1}^\delta\| \leq \max\{\tilde{L}_k, c_{uq}, 1\} = \tilde{L}_{max} \quad (3.43)$$

There are still four important terms which include the matrix  $Y_{k,i-1}^\delta$  and need to be bounded. The first term is  $\|I - \mathbf{H}_{k,i-1}^\delta\|$ , while the remaining ones are the first, second, and the fourth moments of  $Y_{k,i-1}^\delta s_{k,i}^\delta(\psi_{k,i-1})$ . In Appendix A.3, we need an upper bound for the term  $\|I - \mathbf{H}_{k,i-1}^\delta\|$ , and we present its proof here as

follows

$$\begin{aligned}
& \|I - \mathbf{H}'_{k,i-1}{}^\delta\| \stackrel{a}{=} \|I - Y_{k,i-1}^\delta H_{k,i-1}\| \\
& = \|Y_{k,i-1}^\delta ((Y_{k,i-1}^\delta)^{-1} - H_{k,i-1})\| \\
& \stackrel{b}{\leq} \|Y_{k,i-1}^\delta\| \|((Y_{k,i-1}^\delta)^{-1} - H_{k,i-1})\| \\
& \stackrel{c}{\leq} \|Y_{k,i-1}^\delta\| \|(\nabla^2 J_k(\psi_{k,i-1}) - H_{k,i-1})\| \\
& \stackrel{d}{\leq} \|Y_{k,i-1}^\delta\| * \left\| \int_0^1 \nabla^2 J_k(\psi_{k,i-1}) - \nabla^2 J_k(\omega^o - t\tilde{\psi}_{k,i-1}) dt \right\| \\
& \stackrel{e}{\leq} L_{max} * \int_0^1 \|\nabla^2 J_k(\psi_{k,i-1}) - \nabla^2 J_k(\omega^o - t\tilde{\psi}_{k,i-1})\| dt \\
& \stackrel{f}{\leq} L_{max} \int_0^1 \mathcal{K}'_d \|\psi_{k,i-1} - \omega^o + t\tilde{\psi}_{k,i-1}\| dt \\
& = L_{max} \mathcal{K}'_d \int_0^1 \|\tilde{\psi}_{k,i-1}(t-1)\| dt \\
& = L_{max} \mathcal{K}'_d C \|\tilde{\psi}_{k,i-1}\| \\
& = L_{max} \mathcal{K}'_d C \left\| \sum_{l \in \mathcal{N}_k} \tilde{\omega}_{l,i-1} \right\| \\
& \leq L_{max} \mathcal{K}'_d C N \|\tilde{\omega}_{i-1}\|
\end{aligned} \tag{3.44}$$

where (a), (b), (c), (d), (e) and (f) follow from (3.26), triangular inequality, (3.4), mean-value theorem, triangular inequality and Lipschitz condition, while  $C$  is some positive constant.

Now, regarding the moments of  $Y_{k,i-1}^\delta s_{k,i}^{\delta=SG}(\psi_{k,i-1})$ , it is useful to recall that the gradient noise is non-zero only when the distributed agents are applying the SGD. The gradient noise in (3.6) plays an important role in the stability analysis as will be shown in Section 3.3, and the following is an important assumption about the gradient noise for agents who apply the SGD.

### Assumption 2 Moments of Gradient Noise

The gradient noise process  $s_{k,i}^{\delta=SG}(\psi_{k,i-1})$  has non-zero value over agents who implement the SGD as given in (3.6). In addition, their matrix  $Y_{k,i-1}^{\delta=SG} = I_M$  as given in (3.39). Then, the moments of the gradient noise in [2] remain unchanged for the term  $Y_{k,i-1}^{\delta=SG} s_{k,i}^{\delta=SG}(\psi_{k,i-1})$  and thus for any  $\omega \in \mathcal{F}_{i-1}$  and for all  $k = 1, 2, \dots, N$ , we have

$$\mathbf{E}[Y_{k,i-1}^{\delta=SG} s_{k,i}^{\delta=SG}(\psi_{k,i-1}) | \mathcal{F}_{i-1}] = 0 \tag{3.45}$$

$$\mathbf{E}[\|Y_{k,i-1}^{\delta=SG} s_{k,i}^{\delta=SG}(\psi_{k,i-1})\|^2 | \mathcal{F}_{i-1}] \leq \beta_k^2 \|\tilde{\psi}_{k,i-1}\|^2 + \sigma_{s,k}^2 \tag{3.46}$$

$$\mathbf{E}[Y_{k,i-1}^{\delta=SG} s_{k,i}^{\delta=SG}(\psi_{k,i-1}) (s_{l,i}^{\delta=SG})^T(\psi_{k,i-1}) | \mathcal{F}_{i-1}] = 0, \quad k \neq l \tag{3.47}$$

$$\mathbf{E}[\|s_{k,i}^{\delta=SG}(\psi_{k,i-1})\|^4 | \mathcal{F}_{i-1}] \leq \beta_k^4 \|\tilde{\psi}_{k,i-1}\|^4 + \sigma_{s,k}^4 \tag{3.48}$$

for some  $\beta_k^2 \geq 0$ ,  $\sigma_{s,k}^2 \geq 0$ , and  $\mathcal{F}_{i-1}$  represents the set of the previous iterates  $\{\omega_{i-1}\}$ . ■

As explained in [2], these conditions are automatically satisfied in many circumstances of interest in learning and adaptation. Condition (3.45) states that the gradient noise is unbiased conditioned on the past iterates. Condition (3.46) states that the second-order moment of the gradient noise is bounded by the squared norm of the iterates. Condition (3.47) shows that the gradient noises across the agents are uncorrelated.

### 3.2.6 Analysis of the Matrix $\mathcal{B}_{i-1}$

Matrix  $\mathcal{B}_{i-1}$  plays an important role in the stability and the convergence analysis of (3.16). In this section, we will present a decomposition of the matrix  $A_1$  since it facilitates the proof of several upper bounds for different elements within the equation of the matrix  $\mathcal{B}_{i-1}$ .  $A_1$  is a left stochastic matrix and assumed to be primitive. The Jordan canonical decomposition of the  $N \times N$  matrix  $A_1$  is given by

$$A_1 = V_\epsilon J V_\epsilon^{-1} \quad (3.49)$$

where  $J$ ,  $V_\epsilon$ , and  $V_\epsilon^{-1}$  have dimensions of  $N \times N$  and they are given by

$$J = \begin{pmatrix} 1 & 0 \\ 0 & J_\epsilon \end{pmatrix} \quad (3.50)$$

$$V_\epsilon = \begin{pmatrix} p & V_R \end{pmatrix} \quad (3.51)$$

$$V_\epsilon^{-1} = \begin{pmatrix} 1^T \\ V_L^T \end{pmatrix} \quad (3.52)$$

where the matrices  $V_L$ ,  $J_\epsilon$ ,  $V_R$  have dimensions of  $(N-1) \times (N-1)$ . Since  $V_\epsilon^{-1} V_\epsilon = I_N$ , then it holds that,

$$\begin{cases} 1^T V_R = \mathbf{0}^T \\ V_L^T p = \mathbf{0} \\ V_L^T V_R = I_{N-1} \end{cases} \quad (3.53)$$

Substituting (3.49) into (3.18), we get

$$\mathcal{B}_{i-1} = ((V_\epsilon^{-1})^T \otimes I_M) \{ (J^T \otimes I_M) - \mathcal{D}_{i-1}^T \} (V_\epsilon^T \otimes I_M) \quad (3.54)$$

where  $\mathcal{D}_{i-1}^T$  is given by

$$\begin{aligned} \mathcal{D}_{i-1}^T &= ((V_\epsilon)^T \otimes I_M) \mathcal{M} \mathcal{H}'_{i-1} \mathcal{A}_1^T ((V_\epsilon^{-1})^T \otimes I_M) \\ &= \begin{pmatrix} \mathbf{D}_{11,i-1}^T & \mathbf{D}_{21,i-1}^T \\ \mathbf{D}_{12,i-1}^T & \mathbf{D}_{22,i-1}^T \end{pmatrix} \end{aligned} \quad (3.55)$$

The given form of  $\mathcal{B}_{i-1}$  in (3.54) is the one that will be used for proving the stability in the next section. However, it is still remaining in this section to find the upper bounds of the terms  $\|\mathbf{D}_{11,i-1}^T\|$ ,  $\|\mathbf{D}_{12,i-1}^T\|$ ,  $\|\mathbf{D}_{21,i-1}^T\|$ , and  $\|\mathbf{D}_{22,i-1}^T\|$  of the matrix  $\mathcal{B}_{i-1}$ . Substituting (3.51), (3.52) and (3.53) in (3.55), we get

$$\mathbf{D}_{11,i-1} = \sum_{k=1}^N \mu_k p_k \mathbf{H}'_{k\delta,i-1}{}^T \quad (3.56)$$

$$\mathbf{D}_{12,i-1} = (\mathbf{1}^T \otimes I_M) \mathcal{H}_{i-1}{}^T \mathcal{M}(V_R \otimes I_M) \quad (3.57)$$

$$\mathbf{D}_{21,i-1} = (V_L^T A_1 \otimes I_M) \mathcal{H}_{i-1}{}^T \mathcal{M}(p \otimes I_M) \quad (3.58)$$

$$\mathbf{D}_{22,i-1} = (V_L^T A_1 \otimes I_M) \mathcal{H}_{i-1}{}^T \mathcal{M}(V_R \otimes I_M) \quad (3.59)$$

Let  $\mu_k$  be defined as

$$\mu_k = \tau_k \mu_{max} \quad (3.60)$$

where  $\tau_k$  is some positive scalar, then the matrix sequence  $\mathbf{D}_{11,i-1}$  can be upper bounded as follows

$$\begin{aligned} \mathbf{D}_{11,i-1} &= \sum_{k=1}^N \mu_k p_k Y_{k,i-1}^\delta \mathbf{H}_{k,i-1}{}^{\delta T} \\ &\stackrel{a}{\leq} p_{max} \tau_{k_{max}} \mu_{max} N c_2' \delta_c I_M \\ &= c_2'' \mu_{max} I_M = O(\mu_{max}) \end{aligned} \quad (3.61)$$

where (a) follows from (3.13) and (3.42),  $\mu_{max}$  and  $p_{max}$  represent the maximum learning rate and the maximum entry of the vector  $p$  respectively. The lower bound of  $\mathbf{D}_{11,i-1}$  can be derived as follows

$$\begin{aligned} \mathbf{D}_{11,i-1} &= \sum_{k=1}^N \mu_k p_k Y_{k,i-1}^\delta \mathbf{H}_{k,i-1}{}^{\delta T} \\ &\stackrel{a}{\geq} c_1' p_{k_o} \tau_{k_o} \mu_{max} \nu I_M \\ &= c_1'' \mu_{max} I_M = O(\mu_{max}) \end{aligned} \quad (3.62)$$

where (a) follows from (3.14) and (3.32).

In the following sections, we need to know the induced 2-norm of  $\|I_M - \mathbf{D}_{11,i-1}^T\|$  and this can be derived as follows. We can deduce from (3.61), and (3.62) that

$$\begin{aligned} c_1'' \mu_{max} I_M &\leq \mathbf{D}_{11,i-1} \leq c_2'' \mu_{max} I_M \\ -c_2'' \mu_{max} I_M &\leq -\mathbf{D}_{11,i-1} \leq -c_1'' \mu_{max} I_M \\ (1 - c_2'' \mu_{max}) I_M &\leq I_M - \mathbf{D}_{11,i-1} \leq (1 - c_1'' \mu_{max}) I_M \\ \|I_M - \mathbf{D}_{11,i-1}^T\| &\leq \max\{1 - c_1'' \mu_{max}, 1 - c_2'' \mu_{max}\} \\ &= 1 - O(\mu_{max}) \end{aligned} \quad (3.63)$$



$\|\mathbf{D}_{21,i-1}\|$  can be bounded as follows

$$\begin{aligned}
\|\mathbf{D}_{21,i-1}\| &= \|(V_L^T A_1 \otimes I_M) \mathcal{H}_{i-1}^{T'} \mathcal{M}(p \otimes I_M)\| \\
&\leq \|(V_L^T A_1 \otimes I_M)\| \|\mathcal{H}_{i-1}^{T'}\| \|\mathcal{M}\| \|p \otimes I_M\| \\
&= \|(V_L^T A_1 \otimes I_M)\| \|\mathcal{H}_{i-1}^{T'}\| \|\mathcal{M}\| \sqrt{N p_{max}^2} \\
&= \alpha_{21} \|\mathcal{H}_{i-1}^{T'}\| \|\mathcal{M}\| \\
&\stackrel{a}{\leq} \alpha_{21} \|\mathcal{M}\| (\max_{1 \leq k \leq N} \|\mathbf{H}_{k,i-1}^{T\delta}\|) \\
&\stackrel{b}{\leq} \alpha_{21} \|\mathcal{M}\| \tilde{L}_{max} \delta_c \\
&= \alpha_{21} \|\mathcal{M}\| \tilde{L}_{max} \delta_c \\
&\leq \sigma_{21} \mu_{max} = O(\mu_{max})
\end{aligned} \tag{3.64}$$

where  $\sigma_{21}$  is some positive constant, (a) follows from the property of the induced 2-norm of block diagonal matrices, and (b) follows from (3.13), and (3.43). Similarly, it can be shown that

$$\begin{aligned}
\|\mathbf{D}_{12,i-1}\| &\leq \sigma_{12} \mu_{max} = O(\mu_{max}) \\
\|\mathbf{D}_{22,i-1}\| &\leq \sigma_{22} \mu_{max} = O(\mu_{max})
\end{aligned} \tag{3.65}$$

for some constants  $\sigma_{12}$  and  $\sigma_{22}$ .

### 3.3 Stability Analysis

Using the results of the previous section, we examine the stability of the mean-error process  $\|\mathbf{E} \tilde{\omega}_i^\delta\|$ , the mean-square-error  $\mathbf{E} \|\tilde{\omega}_i^\delta\|^2$  and the fourth-order moment  $\mathbf{E} \|\tilde{\omega}_i^\delta\|^4$  of the distributed strategy in (3.16). It is important to highlight that we need the stability analysis of the heterogeneous algorithms that are being run over the network to guarantee that fast algorithms don't lead to diverge the slow ones and vice versa. In this section, we study how well the distributed strategies in (3.15) and (3.16) approach the optimal solution  $\omega^o$  of the global cost in (2.2). This can be measured by the first, second and fourth order moments of the error between the iterate  $\omega_{k,i}$  and the optimal point  $\omega^o$ . The following theorems are proved in this work.

**Theorem 1 (Network Mean-Square-Error Stability)**

Assume that the aggregate cost  $J^{glob}(\omega)$ , the individual costs  $J_k(\omega)$ , and the network topology satisfy the conditions in **Assumption 1** and **Assumption 2**. Then, the network with heterogeneous agents is mean-square stable for sufficiently small step-sizes, specifically, it holds that

$$\limsup_{i \rightarrow \infty} \mathbf{E} \|\tilde{\omega}_{k,i}\|^2 = O(\mu_{max}), \text{ for } k = 1, 2, \dots, N \quad (3.66)$$

for any  $\mu_{max} < \mu_o$ , for some small enough  $\mu_o$ .

*Proof:* See Appendix A.1.

**Theorem 2 (Network Fourth-Order Moment Stability)**

Assume that the aggregate cost  $J^{glob}(\omega)$ , the individual costs  $J_k(\omega)$ , and the network topology satisfy the conditions in **Assumption 1** and **Assumption 2**. Then, the fourth-order moment,  $\mathbf{E} \|\tilde{\omega}_{k,i}\|^4$ , of the network with heterogeneous agents is stable for sufficiently small step-sizes, specifically, it holds that

$$\limsup_{i \rightarrow \infty} \mathbf{E} \|\tilde{\omega}_{k,i}\|^4 = O(\mu_{max}^2), \text{ for } k = 1, 2, \dots, N \quad (3.67)$$

for any  $\mu_{max} < \mu_o$ , for some small enough  $\mu_o$ .

*Proof:* See Appendix A.2.

**Theorem 3 (Network Mean-Error Stability)**

Assume that the aggregate cost  $J^{glob}(\omega)$ , the individual costs  $J_k(\omega)$ , and the network topology satisfy the conditions in **Assumption 1** and **Assumption 2**. Then, the first-order moment,  $\|\mathbf{E} \tilde{\omega}_{k,i}\|$ , of the network with heterogeneous agents are stable for sufficiently small step-sizes, specifically, it holds that

$$\limsup_{i \rightarrow \infty} \|\mathbf{E} \tilde{\omega}_{k,i}\| = O(\mu_{max}), \text{ for } k = 1, 2, \dots, N \quad (3.68)$$

for any  $\mu_{max} < \mu_o$ , for some small enough  $\mu_o$ .

*Proof:* See Appendix A.3.

## 3.4 Simulation Results and Performance Analysis

In this section, we illustrate the performance of the optimization algorithms for learning over distributed agents who own heterogeneous computational capabilities and can run different iterative optimization algorithms. The performance of

the proposed work is compared with existing algorithms that assume the homogeneous computational capabilities among the dispersed agents.

Figure 4.3 presents an example of a strongly connected network of  $N = 10$  agents. Every agent is assumed to own a regularized logistic cost function  $J_k(\omega)$  that is given by

$$J_k(\omega_k) = \frac{\rho}{2} \|\omega_k\|^2 + \frac{1}{L} \sum_{n=0}^{L-1} \ln(1 + e^{-\gamma_k(n)(h_k(n))^T \omega_k}) \quad (3.69)$$

The gradient and the Hessian matrix of (3.69) are given in (3.70), and (3.71) respectively.

$$\nabla J_k(\omega_k) = \rho \omega_k - \frac{1}{L} \sum_{n=0}^{L-1} \gamma_k(n) h_k(n) \frac{e^{-\gamma_k(n)(h_k(n))^T \omega_k}}{1 + e^{-\gamma_k(n)(h_k(n))^T \omega_k}} \quad (3.70)$$

$$\nabla^2 J_k(\omega_k) = \rho I_M + \frac{1}{L} \sum_{n=0}^{L-1} h_k(n)(h_k(n))^T \frac{e^{-\gamma_k(n)h_k(n)^T \omega_k}}{(1 + e^{-\gamma_k(n)h_k(n)^T \omega_k})^2} \quad (3.71)$$

where  $L$ ,  $\gamma_k(n) \in R$ ,  $h_k(n) \in R^M$  ( $M = 50$ ) and  $\rho$  represent the size of the dataset, the label, the data and the regularization parameter respectively. Agents with low computational capabilities (when  $\delta = SG$ ) approximate the gradient in (3.70) by  $\widehat{\nabla J}_k(\omega_k)$  which is given below by

$$\widehat{\nabla J}_k(\omega_k) = \rho \omega_k - (\gamma_k(b) h_k(b)) \frac{e^{-\gamma_k(b) h_k^T(b) \omega_k}}{1 + e^{-\gamma_k(b) h_k^T(b) \omega_k}} \quad (3.72)$$

where  $b$  is the index of the randomly selected pair of data samples  $\{\gamma_k(n), h_k(n)\}$  by the SGD at every iteration.

The objective of every agent is to minimize (3.69) in a distributed manner. All agents are assumed to have the same cost function  $J_k(\omega)$ . We choose  $\alpha_k$  in (3.21) to be equal one for all agents. We generate a random dataset  $\{h_k(n), \gamma_k(n)\}$  of size  $N = 10000$ , where  $h_k(n) \in R^{50}$  and  $\gamma_k(n) \in \pm 1$ . The vectors  $\{h_k(n)\}$  are generated according to a multivariate normal distribution and the corresponding  $\{\gamma_k(n)\}$  values are generated according to Bernoulli distributions with parameter  $h_k(n)^T \omega_k^*$ , where  $\omega_k^*$  is some preset vector. The regularization parameter  $\rho$  is chosen to be equal 0.1.

The step-size  $\mu_k$  is assumed to be uniform ( $\mu_k = \mu$ ), across all agents who implement the SGD and set to be  $10^{-3}$ . For agents who implement the Newton's or the full gradient descent methods, the step-size  $\mu_k$  is chosen by a backtracking line search method based on Armijo condition [17], [18]. The weights of the matrix  $A_1$  are chosen according to the averaging rule as follows

$$a_{lk} = \begin{cases} \frac{1}{|\mathcal{N}_k|}, & \text{if } l \in \mathcal{N}_k \\ 0, & \text{otherwise} \end{cases} \quad (3.73)$$

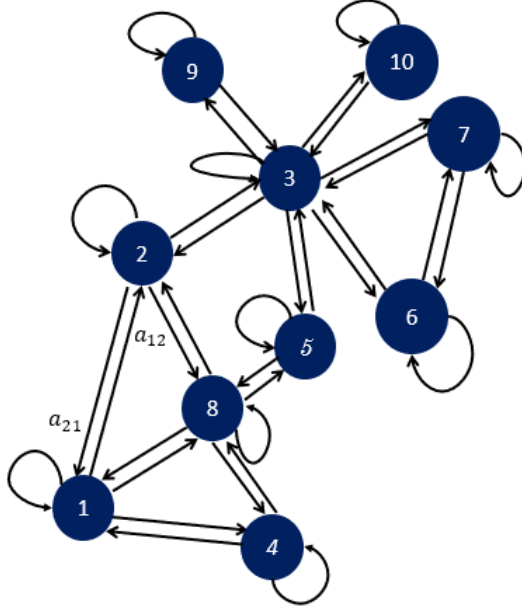


Figure 3.1: A strongly connected network with  $N = 10$  agents who own heterogeneous computational capabilities. Subset of the agents can run the Newton's methods, or the GD, and the remaining ones can only implement the SGD.

where  $|\mathcal{N}_k|$  denotes the degree of the agent  $k$ , which is equal to the size of its neighborhood.

Two different metrics are used in this work to assess the performance of the proposed algorithm. In the first metric, we will measure the mean-square-deviation (MSD) at every agent  $k$ , as well as the MSD for the entire network, by using the following formulas

$$MSD_k = \mathbf{E}\|\tilde{\omega}_{k,i}\|^2 \quad (3.74)$$

$$MSD_{avg} = \frac{1}{N} \sum_{k=1}^N MSD_k \quad (3.75)$$

In the second metric, we will measure the excess risk (ER) at every agent  $k$ , and for the entire network. ER measures the average fluctuation of  $J^{glob}(\omega)$  around its minimum value  $J^{glob}(\omega^o)$ . The ER at every agent  $k$  and for the entire network are defined as follows

$$ER_k = \mathbf{E}\{J^{glob}(\omega_{k,i-1}) - J^{glob}(\omega^o)\} \quad (3.76)$$

$$ER_{avg} = \frac{1}{N} \sum_{k=1}^N ER_k \quad (3.77)$$

### 3.4.1 Impact on Agents with High Computational Capabilities

The proposed distributed optimization for learning over networks has an imperative impact on both capable and non-capable agents. The convergence rate of agents with low computational capabilities will increase after its cooperation with agents with high computational capabilities, however the capable agents will undergo a decrease in their rate of convergence. To assess the impact of the proposed work on the capable agents, we will study four different scenarios in this section; the first one is when all agents over the network can implement the Newton's method, the second one is when agents three and eight can implement the Newton's method while the agents in their neighborhood can only implement the SGD, the third scenario is when all agents over the network can implement the GD method, and the fourth case is when agents three and eight can implement the GD method while the agents in their neighborhood can only implement the SGD. Figure 3.2 presents the normalized curves for agents three and eight under the aforementioned four scenarios. The curves are attained by averaging the trajectories of agents three and eight over 200 repeated experiments. Each experiment involves running the distributed strategy in (3.2) with  $\rho = 0.1$ . The unknown vector  $\omega_{k,i-1}$  is generated randomly at iteration  $i = 0$ , and its norm is normalized to one. Note that the curves of Figure 3.2 are normalized and they are attained by applying (3.75) for capable agents only, i.e.,  $MSD_{avg} = \frac{1}{2}(MSD_3 + MSD_8)$ . The following conclusions can be deduced from Figure 3.2.

- The learning curves over capable agents converge to a stationary point and attain a stable solution, which proves that the cooperation with nodes who implement the SGD does not affect the stability of Newton's or GD descent methods.
- The convergence rate over agents who can implement the Newton's or GD method is degraded after its cooperation with agents who are running the SGD. When all agents run the Newton's method, the true optimal point is obtained with a few number of iterations and this is expected due to the quadratic convergence of the Newton's method. However, this rate of convergence is being affected when the capable agents cooperate with distributed nodes who implement the SGD. This degradation in the rate of convergence is still acceptable as it contributes to improving the speed of the SGD algorithm which is used by most nodes over the network.

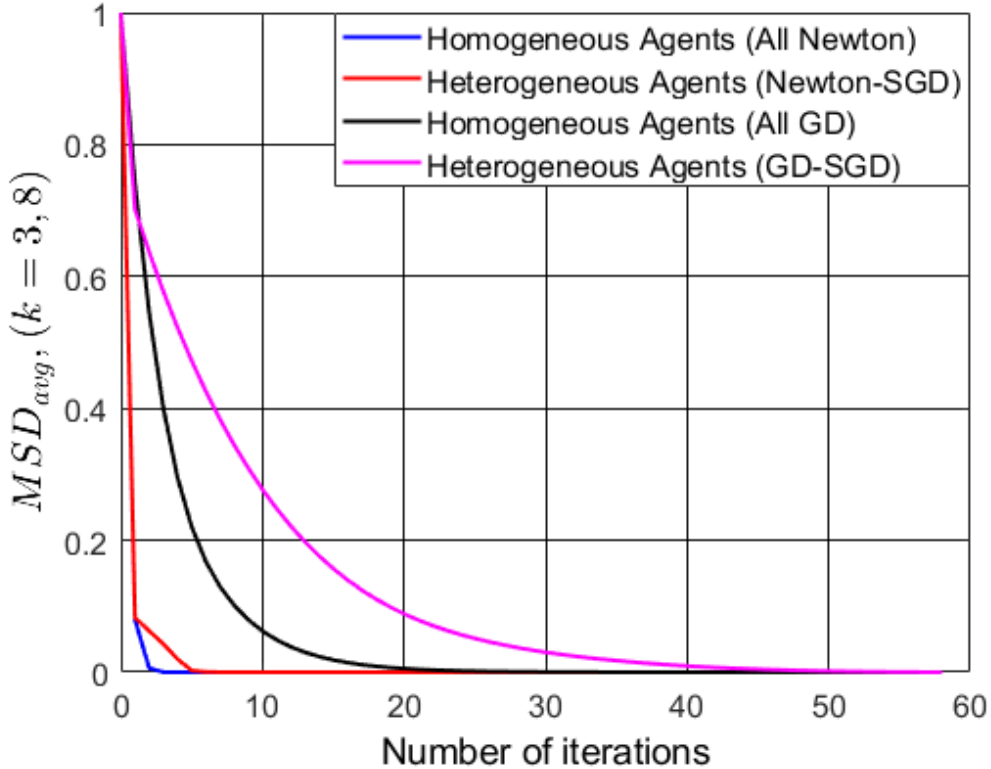


Figure 3.2: Impact of heterogeneous distributed optimization algorithms on the capable agents.

### 3.4.2 Impact on Agents with Low Computational Capabilities

In this section, we study the impact of learning among heterogeneous agents on the nodes that own low computational capabilities. Three different scenarios will be studied over the network of Figure 4.3; the first case is when two of the agents (agents three and eight) can implement the Newton’s method while the remaining ones are running the SGD, the second one is when two of the agents (agents three and eight) can implement the GD method while the remaining ones are running the SGD, and the third case is when all the agents over the network are running the SGD method. Figures 3.3, 3.4 present the normalized curves for non-capable agents (all agents except three and eight) under the aforementioned three scenarios. Figure 3.3 presents the evolution of the ensemble-average learning curves  $MSD_{avg}$  for the distributed learning method given in (3.2). The curves are attained by averaging the trajectories  $MSD_{avg}$  over 200 repeated experiments. Each experiment involves running the distributed strategy in (3.2) with  $\rho = 0.1$ . The unknown vector  $\omega_{k,i-1}$  is generated randomly at iteration  $i = 0$ , and its

norm is normalized to one. Note that the red and blue curves in Figure 3.3 correspond to the ensemble-average learning curves  $MSD_{avg}$  over agents who are implementing the SGD (all agents except three and eight). The following conclusions can be deduced from Figure 3.3.

- Optimization and learning over networks with heterogeneous agents converge to a stationary point and attain a stable solution. Not only this, but the convergence rate is much faster than learning over networks with homogeneous agents who are running SGD. For instance, the hybrid learning of the Newton's method and the SGD requires 11 iterations before convergence, however when all agents are running the SGD then 500 iterations are required before convergence.
- The reduced number of iterations has an imperative impact on reducing the communication overhead over the network and in providing an efficient solution to reduce the transmitted power during the learning process.

Figure 3.4 presents the evolution of the ensemble-average learning curves  $ER_{avg}$  for the distributed learning method given in (3.2). The curves are attained by averaging the trajectories  $ER_{avg}$  over 200 repeated experiments. Each experiment involves running the distributed strategy in (3.2). Note that the red and blue curves correspond to the ensemble-average learning curves  $ER_{avg}$  over agents who are implementing the SGD (all agents except agents three and eight). The following conclusions can be deduced from Figure 3.4.

- The cost functions of agents who implement the SGD attain their minimizers at faster rate of convergence under heterogeneous learning than homogeneous learning.
- The performance of learning among heterogeneous agents is better than the performance of learning among homogeneous agents who implement the SGD. This is clear if we notice that the heterogeneous agents converge faster than homogeneous agents do.

In conclusion, learning over networks with heterogeneous nodes can help agents with limited computational capabilities to increase their rate of convergence and to have a stable solution. The advantage of applying SGD lies in its low computational complexity at every iteration, however, it needs more iterations to converge in comparison with full gradient descent or Newton's method. Hence, allowing agents who implement SGD to cooperate with another ones who run fast algorithms like Newton's methods would lead in getting better performance and faster rate of convergence for agents with limited computational capabilities. Enhancing the rate of convergence at agents with SGD leads to decreasing the number of iterations which in turn reduces the communication overhead of

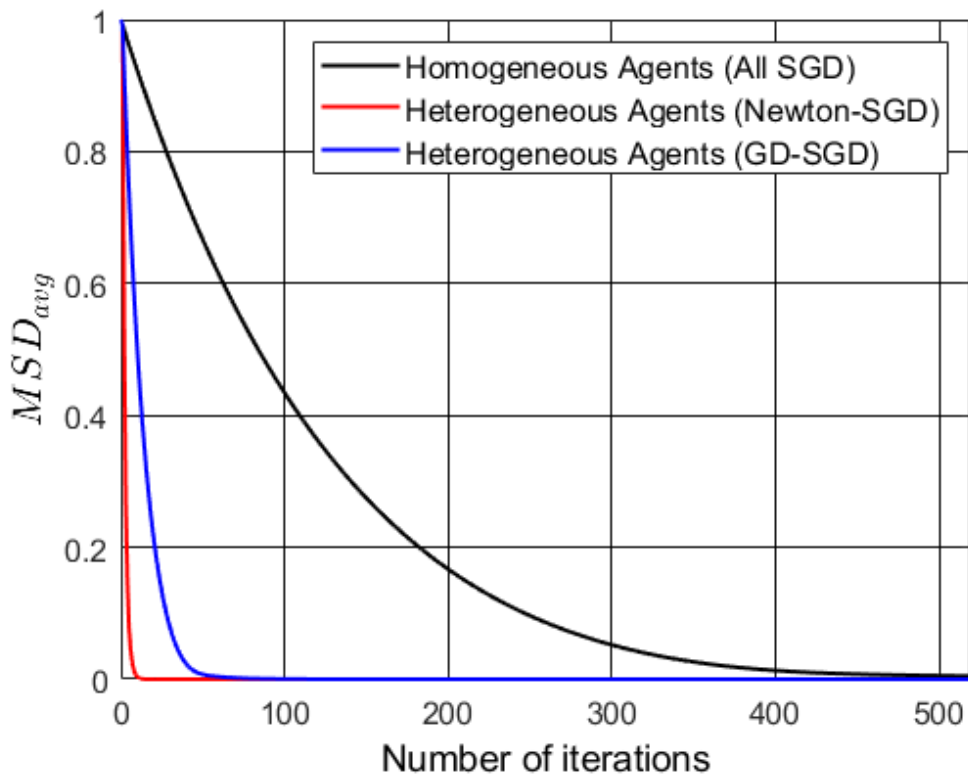


Figure 3.3: Evolution of the mean-square deviation learning curves over network with agents of homogeneous and heterogeneous computational capabilities under the diffusion learning strategy. Agents with limited computational capabilities employ the same step-size of  $\mu = 10^{-3}$ .



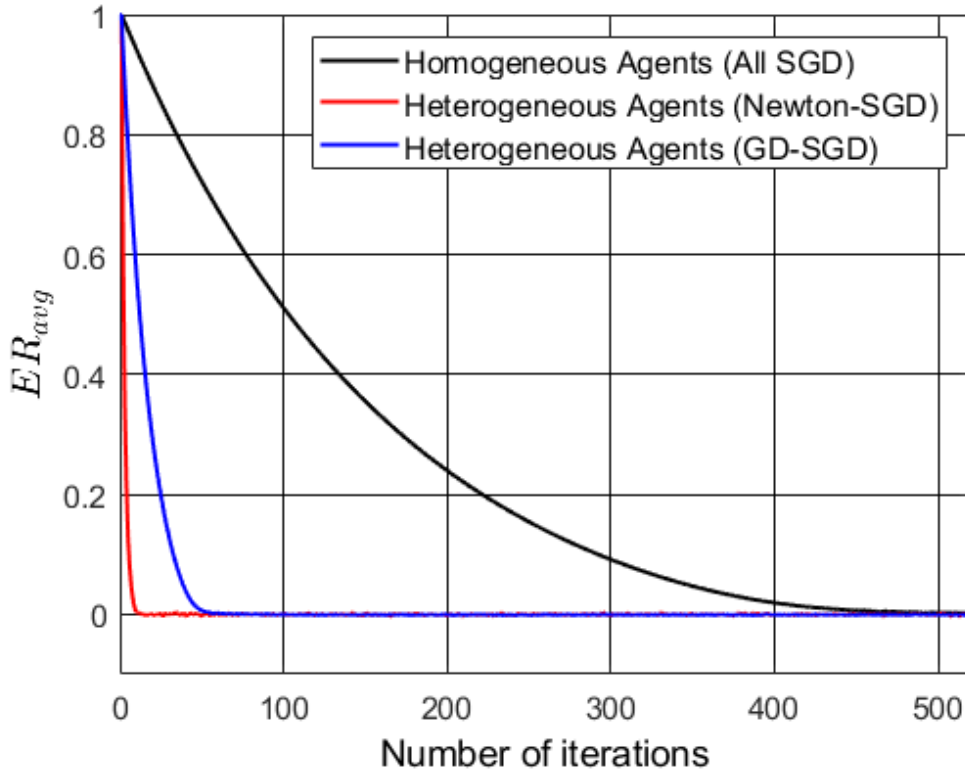


Figure 3.4: Evolution of the empirical risk learning curves over network with agents of homogeneous and heterogeneous computational capabilities under the diffusion learning strategy. Agents with limited computational capabilities employ the same step-size of  $\mu = 10^{-3}$ .

parameters over the network at every iteration. In addition, fast rates of convergence with low computational complexity at every iteration reduce the power consumption at these agents.

### 3.5 Conclusion

This part of the thesis considered the distributed optimization algorithms for learning over dispersed agents who own heterogeneous computational capabilities. We proposed an iterative and distributed implementation that allows subset of the agents to run the SGD while the other capable agents can run the Newton’s method, full gradient descent, etc. Theoretical and simulation results show that, for small step-size parameter  $\mu$ , the distributed learning algorithm is stable and converges faster than the algorithms that work over agents with homogeneous computational capabilities.

# Chapter 4

## A Low Complexity Approximation of Gradient Descent for Learning over Single and Multi-Agent Systems

### 4.1 Background and Literature Review

Learning over networks induces many degrees of freedom, and there are many factors that can improve the performance, increase the rate of convergence, reduce the local power consumption, and enhance the communication overhead over the network. In this part of the thesis, we provide a new approach for efficient approximation of the gradient descent and study its impact on learning over single and multi-agent systems. Most prior literature focuses on distributed algorithms that assume that the dispersed agents apply the stochastic gradient descent (SGD) or the mini-batch learning algorithms due to their low computational complexity in comparison with gradient descent (GD) or Newton's methods. For instance, the methods of incremental [2], consensus [2], [3], diffusion [2], [4], [5] are among the existing distributed strategies that assume that the agents are running the *SGD*. However, these algorithms take time to converge and trade-off the performance with low computational complexity at every iteration. This work aims to approximate the *GD* and study its impact on the convergence rate and the performance of learning algorithms over single and multi-agent systems.

The main contribution in this part is in proposing a low complexity approximation method for the *GD* that has better convergence rate and performance than existing methods. The theoretical and simulation results show that the proposed method converges to a stationary point, improve the rate of convergence and the performance of learning over single and multi-agent systems. Improving the rate of convergence leads to reducing the local power consumption, decreasing

the communication overhead and saving costs resulting from running the network for a long time.

## 4.2 Proposed Approach for Approximating the Gradient Descent Algorithm

Consider the optimization problem of minimizing over model parameters  $\omega \in \mathcal{R}^M$  the cost function  $P(\omega, \{h(i), \gamma(i)\}_{i=1}^N)$ , defined over a dataset of  $N$  data samples  $h \in \mathcal{R}^M$  and their respective labels  $\gamma \in \{0, 1, \dots, C - 1\}$ . The cost function  $P(\omega)$  is assumed to be differentiable over  $\omega$  and can be expressed as a sum of loss function  $Q$  over individual data samples as

$$P(\omega) = \frac{1}{N} \sum_{i=0}^{N-1} Q(\omega, h(i), \gamma(i)) \quad (4.1)$$

Batch gradient descent (GD) [17] provides a solution to the minimization of  $P(\omega)$  by iteratively updating the parameter vector  $\omega$  in the direction opposite to the gradient vector  $\nabla P(\omega)$  at steps of  $\mu$

$$\text{GD: } \omega_n = \omega_{n-1} - \mu \nabla P(\omega_{n-1}) \quad (4.2)$$

By linearity of the differentiation operator and (4.1), the computation of the gradient  $\nabla P(\omega)$  in (4.2) comprises  $N$  computations of gradients  $\nabla Q(\omega)$  corresponding to the  $N$  data samples. For large datasets, such a computation may be cost prohibitive. Alternatively, mini-batch gradient descent updates parameter vector  $\omega$  per iteration  $n$  at the cost of  $B$  computations of gradients  $\nabla Q(\omega)$  over a random selection  $\mathcal{S}_n$  of  $B$  data samples of the dataset

$$\begin{aligned} \text{Mini-GD: } \omega_n &= \omega_{n-1} - \mu \nabla \hat{P}(\omega_{n-1}) \\ \hat{P}(\omega) &= \frac{1}{B} \sum_{i \in \mathcal{S}_n} Q(\omega, h(i), \gamma(i)) \end{aligned} \quad (4.3)$$

Assuming that selection  $\mathcal{S}_n$  may include each sample  $(h(i), \gamma(i))$  of the dataset with equal probability  $1/N$ , the expected value  $E[\hat{P}(\omega)]$  of  $\hat{P}(\omega)$  is given by

$$\begin{aligned} E_{\mathcal{S}_n} [\hat{P}(\omega)] &= \frac{1}{B} \sum_{i \in \mathcal{S}_n} E_{(h(i), \gamma(i))} [Q(\omega, h(i), \gamma(i))] \\ &= \frac{1}{B} \sum_{i \in \mathcal{S}_n} \frac{1}{N} \sum_{j=0}^{N-1} Q(\omega, h(j), \gamma(j)) \\ &= \frac{1}{B} \times |\mathcal{S}_n| \times P(\omega) = P(\omega) \end{aligned} \quad (4.4)$$

(4.4) implies that the mini-batch gradient descent algorithm in (4.3) computes that update of parameter vector  $\omega$  using a random approximation  $\widehat{P}(\omega)$  of the cost function  $P(\omega)$  per iteration  $n$ . A tradeoff exists between how well  $-\nabla\widehat{P}(\omega)$  fits the steepest descent direction  $-\nabla P(\omega)$  and how many gradients  $\nabla Q(\omega)$  are computed per iteration.

Using the same insight as for mini-batch gradient descent, we propose another approximation  $\bar{P}(\omega)$  of  $P(\omega)$  that reduces the computational burden of the parameter vector update per iteration. The approximation is based on *deterministically* mapping each data sample  $h(i)$  to a new value  $\bar{h}(i) \in \mathcal{R}^M$  and then computing the gradient descent direction over the new set of samples  $\{\bar{h}(i)\}_{i=1}^N$ . Respective labels  $\{\gamma(i)\}_{i=1}^N$  are unaltered.

$$\begin{aligned} \text{Prop-GD: } \omega_n &= \omega_{n-1} - \mu \nabla \bar{P}(\omega_{n-1}) \\ \bar{P}(\omega) &= \frac{1}{N} \sum_{i=0}^{N-1} Q(\omega, \bar{h}(i), \gamma(i)) \end{aligned} \quad (4.5)$$

The new samples  $\{\bar{h}(i)\}$  are computed such that

- The sum of squares of distances  $\{d(h(i), \bar{h}(i))\}$  between the original and new samples is minimized
- The new samples  $\{\bar{h}(i)\}$  attain at most  $K$  distinct values  $\eta_{1,\gamma(i)}, \dots, \eta_{K,\gamma(i)}$  per class label  $\gamma(i)$

$$\left\{ \begin{aligned} \bar{h}(i) &= f_{\gamma(i)}(h(i)) = \arg \min_{x \in \{\eta_{1,\gamma(i)}, \dots, \eta_{K,\gamma(i)}\}} d(h(i), x)^2 \\ (\eta_{1,\gamma(i)}, \dots, \eta_{K,\gamma(i)}) &= \arg \min_{(\eta_1, \dots, \eta_K)} \sum_{k=1}^K \sum_{\substack{j=0, \\ \bar{h}(j)=\eta_k, \\ \gamma(j)=\gamma(i)}}^{N-1} d(h(j), \eta_k)^2 \end{aligned} \right. \quad (4.6)$$

The summation terms of  $\bar{P}(\omega)$  in (4.5) can be reordered by the class labels  $\{\gamma(i)\}$  and the values  $\{\eta_{k,\gamma(i)}\}$  attained by  $\bar{h}(i)$  in the respective class

$$\begin{aligned} \bar{P}(\omega) &= \frac{1}{N} \sum_{c=0}^{C-1} \sum_{k=1}^K \sum_{\substack{i=0, \\ \bar{h}(i)=\eta_{k,c}, \\ \gamma(i)=c}}^{N-1} Q(\omega, \bar{h}(i), \gamma(i)) \\ &= \frac{1}{N} \sum_{c=0}^{C-1} \sum_{k=1}^K N_{k,c} \times Q(\omega, \eta_{k,c}, c), \\ N_{k,c} &= \sum_{i=0}^{N-1} \mathbf{1}\{\bar{h}(i) = \eta_{k,c}, \gamma(i) = c\} \end{aligned} \quad (4.7)$$

Equation (4.7) shows that the mapping  $h(i) \mapsto \bar{h}(i)$  defined in (4.6) allows to update the parameter vector  $\omega$  in (4.5) per iteration  $n$  at the cost of  $K \times C$  computations of gradients  $\nabla Q(\omega)$  over the new samples  $\{\bar{h}(i)\}$  in comparison with  $N \times C$  computations of gradients  $\nabla Q(\omega)$  over the original data samples when applying the gradient descent algorithm. The latter samples are computed using K-means clustering of the original samples  $\{h(i)\}$  per class label  $\gamma(i)$ . This can be performed only once offline at linear order of complexity  $\mathcal{O}(N)$  [61], [62], where distance  $d(h(i), \bar{h}(i))$  in (4.6) is chosen as the Euclidean norm of the difference between samples  $h(i)$  and  $\bar{h}(i)$ .

### 4.3 Convergence Analysis

Assume  $P(\cdot)$  is  $\nu$ -strongly convex and first-order differentiable in  $\omega$ , then

$$P(\omega_2) \geq P(\omega_1) + \nabla P(\omega_1)^T(\omega_2 - \omega_1) + \frac{\nu}{2} \|\omega_2 - \omega_1\|^2, \forall \omega_1, \omega_2 \quad (4.8)$$

Further assume that  $\nabla P(\cdot)$  is  $\delta$ -Lipschitz [17], [2] in  $\omega$  and  $\nabla Q(\cdot)$   $\delta_d$ -Lipschitz in the data samples:

$$\begin{aligned} \|\nabla P(\omega_2) - \nabla P(\omega_1)\| &\leq \delta \|\omega_2 - \omega_1\|, \forall \omega_1, \omega_2 \\ \|\nabla Q(\omega, h_2, \gamma) - \nabla Q(\omega, h_1, \gamma)\| &\leq \delta_d \|h_2 - h_1\|, \forall h_1, h_2 \end{aligned} \quad (4.9)$$

Define  $\omega^*$  as the global minimizer of  $P(\omega)$  and let

$$\tilde{\omega}_n = \omega^* - \omega_n \quad (4.10)$$

Denote by  $\mathbf{GE}_n^K$  the gradient error upon computing the gradient in (4.5) using data samples  $\{\bar{h}(i)\}$  as computed in (4.6) for a given  $K$  value

$$\mathbf{GE}_n^K = \nabla \bar{P}(\omega_{n-1}) - \nabla P(\omega_{n-1}) \quad (4.11)$$

Using (4.10) and (4.11), the parameter vector update in (4.5) becomes

$$\tilde{\omega}_n = \tilde{\omega}_{n-1} + \mu \nabla P(\omega_{n-1}) + \mu \mathbf{GE}_n^K \quad (4.12)$$

By Cauchy-Schwarz inequality we have

$$\begin{aligned} \|\tilde{\omega}_n\|^2 &\leq (\|\tilde{\omega}_{n-1} + \mu \nabla P(\omega_{n-1})\| + \mu \|\mathbf{GE}_n^K\|)^2 \\ &= \left( \sqrt{g(\omega_{n-1})} + \mu \|\mathbf{GE}_n^K\| \right)^2 \end{aligned} \quad (4.13)$$

Using the results of the Appendix A.4, an upper bound of  $g(\omega_{n-1})$  in (4.13) can be obtained as follows

$$\begin{aligned} g(\omega_{n-1}) &= \|\tilde{\omega}_{n-1}\|^2 + \mu^2 \|\nabla P(\omega_{n-1})\|^2 \\ &\quad + 2\mu \nabla P(\omega_{n-1})^T \tilde{\omega}_{n-1} \\ &\leq \|\tilde{\omega}_{n-1}\|^2 + \mu^2 \delta^2 \|\tilde{\omega}_{n-1}\|^2 - 2\mu\nu \|\tilde{\omega}_{n-1}\|^2 \\ &= (1 - 2\mu\nu + \mu^2 \delta^2) \|\tilde{\omega}_{n-1}\|^2 = \beta \|\tilde{\omega}_{n-1}\|^2 \end{aligned} \quad (4.14)$$

On the other hand, using the triangular inequality and the Lipschitz condition of  $\nabla Q(\cdot)$  in (4.9), an upper bound of  $\|\mathbf{GE}_n^K\|$  in (4.13) can be obtained as follows

$$\begin{aligned}\|\mathbf{GE}_n^K\| &= \left\| \frac{1}{N} \sum_{i=0}^{N-1} \nabla Q(\omega, \bar{h}(i), \gamma(i)) - \nabla Q(\omega, h(i), \gamma(i)) \right\| \\ &\leq \frac{1}{N} \sum_{i=0}^{N-1} \|\nabla Q(\omega, \bar{h}(i), \gamma(i)) - \nabla Q(\omega, h(i), \gamma(i))\| \\ &\leq \frac{\delta_d}{N} \sum_{i=0}^{N-1} d(h(i), \bar{h}(i)) = \delta_d \cdot d_{av}\end{aligned}\tag{4.15}$$

Plugging (4.14) and (4.15) in (4.13), we obtain

$$\|\tilde{\omega}_n\| \leq \beta^{1/2} \|\tilde{\omega}_{n-1}\| + \mu \delta_d \cdot d_{av}\tag{4.16}$$

Taking the limit  $n \rightarrow \infty$  on both sides of (4.16), we have

$$\lim_{n \rightarrow \infty} \sup \|\tilde{\omega}_n\| \leq \frac{\mu \delta_d \cdot d_{av}}{1 - \beta^{1/2}} \leq \frac{\mu \delta_d \cdot d_{av}}{1 - \beta^{1/2}} \Big|_{\mu=0} = \frac{\delta_d \cdot d_{av}}{\nu},\tag{4.17}$$

where the equality in (4.17) follows from applying L'Hopital's rule at  $\mu = 0$ .

The upper bound in (4.17) shows that if  $\bar{h}(i) = h(i) \forall i$ , then  $d_{av} = 0$  and  $\omega_n \xrightarrow[n \rightarrow \infty]{} \omega^*$ . In this case, (4.5) becomes (4.2) and the convergence of batch gradient descent is as expected for strongly convex function  $P(\omega)$ . Otherwise, the selection of  $K$  in (4.6) determines the average distance  $d_{av}$  between data samples  $\{h(i)\}$  and their approximations  $\{\bar{h}(i)\}$ , which in turn determines the upper bound of the convergence radius.

## 4.4 Proposed Method under Distributed Learning

A network of  $R$  agents seek to collaboratively find minimizer  $\omega^o$  of a global cost function  $P^{\text{glob}}(\omega)$  of the form

$$P^{\text{glob}}(\omega) = \sum_{r=1}^R P_r(\omega)\tag{4.18}$$

Each  $P_r(\omega)$  has the same form as (4.1) and is function of dataset  $\{h_r(i), \gamma_r(i)\}_{i=1}^{N_r}$  owned by agent  $r$ . We propose a diffusion-based method to solve for  $\omega^o$ . In

particular, each agent  $r$  runs the following algorithm at every iteration

$$\begin{cases} \psi_{r,n-1} = \sum_{s \in \mathcal{N}_r} a_{sr} \omega_{s,n-1}, \\ \omega_{r,n} = \psi_{r,n-1} - \mu_r \nabla \bar{P}_r(\psi_{r,n-1}) \\ = \psi_{r,n-1} - \mu_r \frac{1}{N_r} \sum_{i=0}^{N_r-1} \nabla Q(\psi_{r,n-1}, \bar{h}_r(i), \gamma_r(i)) \end{cases} \quad (4.19)$$

In the first step of (4.19), every agent  $r$  combines the iterates  $\{\omega_{s,n-1}\}$  received by agent  $r$  from its neighborhood  $\mathcal{N}_r$ . The weights  $\{a_{sr}\}$  are selected by agent  $r$  such that [9–16]

$$a_{sr} \geq 0, \quad \sum_{s=1}^R a_{sr} = 1, \quad \text{and } a_{sr} = 0 \text{ if } s \notin \mathcal{N}_r \quad (4.20)$$

In the second step of (4.19), every agent  $r$  performs an adaptation step based on the computed value  $\psi_{r,n-1}$ . Each agent  $r$  computes the approximation  $\{\bar{h}_r(i)\}$  of its original dataset  $\{h_r(i)\}$  as in (4.6), then it uses it to update the direction  $-\nabla \bar{P}_r(\psi_{r,n-1})$  in the adaptation step. The direction can be computed efficiently as illustrated in (4.7) using a reduced number of gradient computations  $\nabla Q(\psi_r)$  that is dependent on the choice of value  $K$  by agent  $r$ .

## 4.5 Simulation Results and Performance Analysis

Figure 4.3 presents an example of a strongly connected network of  $R = 10$  agents. Every agent  $r$  is assumed to own a regularized logistic cost function  $J_r(\omega_r)$  that is given by

$$J_r(\omega_r) = \frac{\rho}{2} \|\omega_r\|^2 + \frac{1}{N} \sum_{i=0}^{N-1} \ln(1 + e^{-\gamma_r(i)(h_r(i))^T \omega_r}) \quad (4.21)$$

The gradient of (4.21) is given by

$$\nabla J_r(\omega_r) = \rho \omega_r - \frac{1}{N} \sum_{i=0}^{N-1} \gamma_r(i) h_r(i) \frac{e^{-\gamma_r(i)(h_r(i))^T \omega_r}}{1 + e^{-\gamma_r(i)(h_r(i))^T \omega_r}} \quad (4.22)$$

To assess the performance of the proposed algorithm, we measure the mean-square-deviation ( $MSD_r$ ) at every agent  $r$ , as well as the  $MSD_{avg}$  for the entire network as follows

$$MSD_{avg} = \frac{1}{R} \sum_{r=1}^R MSD_r = \frac{1}{R} \sum_{r=1}^R \mathbf{E} \|\tilde{\omega}_{r,n}\|^2 \quad (4.23)$$

In addition, we evaluate the average fluctuation of  $J^{glob}(\omega)$  around its minimum value  $J^{glob}(\omega^o)$  by computing the excess risk ( $ER_r$ ) at every agent  $r$ , and for the entire network ( $ER_{avg}$ ) as follows.

$$ER_r = \mathbf{E}\{J^{glob}(\omega_{r,n-1}) - J^{glob}(\omega^o)\} \quad (4.24)$$

$$ER_{avg} = \frac{1}{R} \sum_{r=1}^R ER_r \quad (4.25)$$

We generate a random dataset  $\{h_r(i), \gamma_r(i)\}$  of size  $N = 10000$ , where  $h_r(i) \in R^{50}$  and  $\gamma_r(i) \in \pm 1$ . The vectors  $\{h_r(i)\}$  are generated according to a multivariate normal distribution and the corresponding  $\{\gamma_r(i)\}$  values are generated according to Bernoulli distributions with parameter  $h_r(i)^T \omega_r^*$ , where  $\omega_r^*$  is some preset vector. We set  $\rho = 0.1$ , and  $\mu_r = 0.01$  for the mini-batch algorithm, while  $\mu_r$  is chosen according to Armijo backtracking line search method [17], [18] for the  $GD$  and the proposed algorithms. The weights  $a_{sr}$  are chosen such that  $a_{sr} = \frac{1}{|\mathcal{N}_r|}$  if  $l \in \mathcal{N}_r$  and zero otherwise. The results of the  $GD$ , mini-batch ( $B = 20$ ) and the proposed (with  $K = 10, 30, 60$ ) learning methods over single and multi-agent systems are presented in Figures 4.1, 4.2, 4.4 and 4.5. The curves are attained by averaging the trajectories of (4.23), (4.24), and (4.25) over 200 repeated experiments. We make the following observations.

- The proposed method converges to a stationary point and attain a stable solution over single and multi-agent systems as shown in Figures 4.1, 4.2, 4.4 and 4.5.
- Figures 4.1 and 4.4 show that the batch gradient takes around 10 iterations to converge and it needs  $N = 10000$  computations of the gradient per iteration, so it requires in total around  $10 * 10000$  computations of the gradient  $\nabla Q$ . However, the proposed algorithm (with  $K = 10$ ) takes around 20 iterations to converge, and it needs  $2 * K = 20$  computations of the gradient at every iteration, thus it needs around  $20 * 20$  computation of the gradient  $\nabla Q$ . This decrease in the required number of iterations has an imperative impact on reducing the local power consumption.
- When the mini-batch algorithm uses  $B = K$ , then it has the same computational complexity at every iteration as the proposed method, however the convergence rate and the steady-state behaviour of the proposed method outperform the mini-batch algorithm as shown in Figures 4.2 and 4.5. The reduced number of iterations in the proposed algorithm in comparison with the mini-Batch method has an imperative impact in reducing the communication overhead over the network and in providing an efficient solution to reduce the transmitted power during the learning process. Note that when  $K = 10$ , then we have 10 clusters in each label of the dataset, and that is



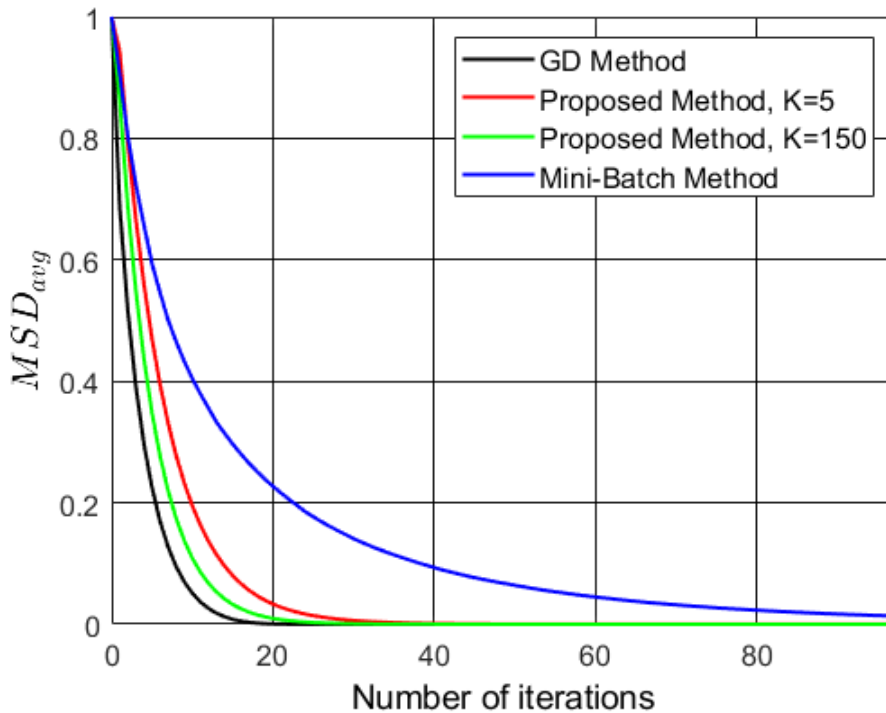


Figure 4.1: Comparison among the mean-square-deviation  $MSD_r$  curves over single agent (agent  $r=1$  in Figure 4.3) of the GD, mini-batch ( $B = 20$ ) and the proposed method ( $K = 10, 30, 60$ ).

why we need a mini-batch with  $B = 20$  to have a fair comparison with the proposed algorithm with  $K = 10$ .

## 4.6 Conclusion

In this part of the thesis, we proposed an approximation method for computing the gradient descent algorithm, and a theoretical convergence analysis is presented. The impact of the proposed method is evaluated over single and multi-agent systems and it outperforms existing algorithms in terms of convergence rate and performance.

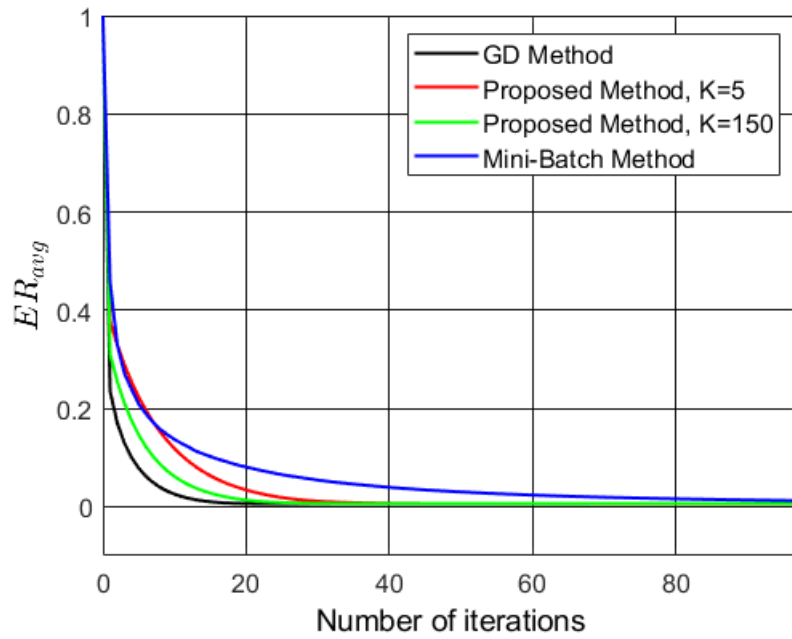


Figure 4.2: Comparison among the excess risk  $ER_r$  curves over single agent (agent  $r=1$  in Figure 4.3) of the GD, mini-batch ( $B = 20$ ) and the proposed method ( $K = 10, 30, 60$ ).

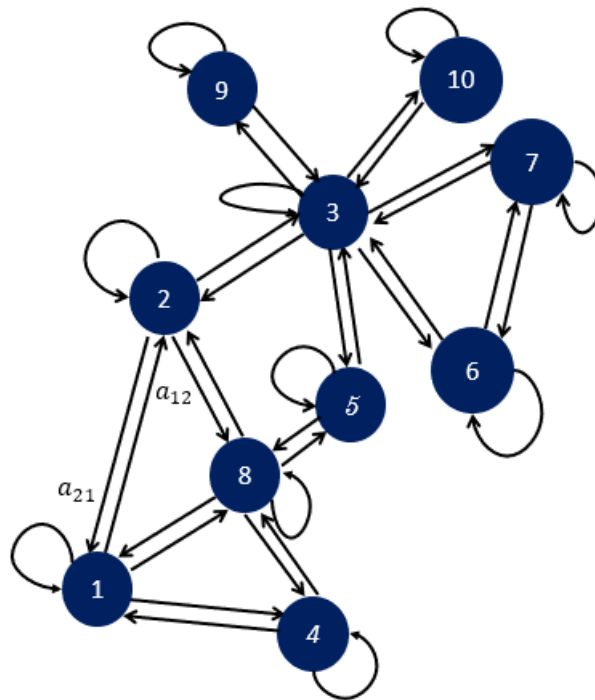


Figure 4.3: A strongly connected network with  $N = 10$  agents.

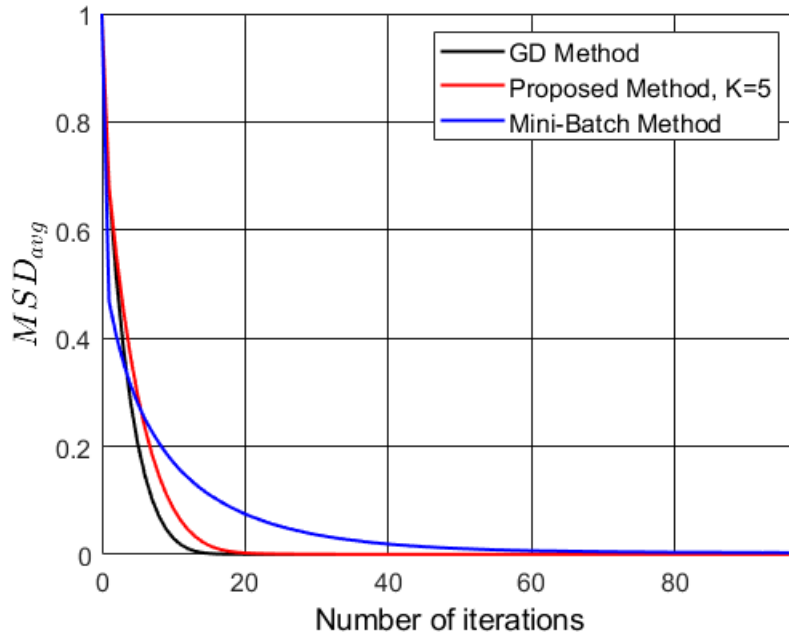


Figure 4.4: Evolution of the mean-square deviation learning curves over network with  $R = 10$  agents under the diffusion learning strategy.

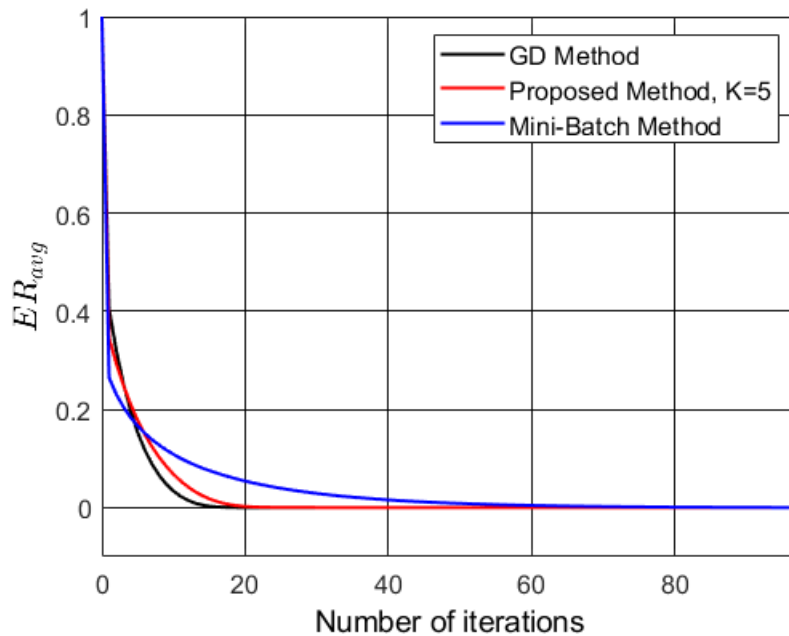


Figure 4.5: Evolution of the excess risks learning curves over network with  $R = 10$  agents under the diffusion learning strategy.

# Chapter 5

## A Clustering-Based Approach for Designing Low Complexity FIR Filters

This is the second part of the thesis, in which we provide a new method of designing fast filtering algorithms for big data and low power DSP applications.

### 5.1 Background and Literature Review

Low power requirements [63–65], real-time constraints [66], and big data applications [67–74] pose significant challenges for the design of efficient signal processing algorithms. The proliferation of smartphones, massive antennas, and dense small cells generates large data sets that need to be processed at edge computing nodes with low delay and high efficiency. The online processing and digital filtering of such data volumes require the design of novel algorithms that are optimized for high speed and low energy consumption [1]. In addition, wearable physiological sensing devices generate large data sets that need to be processed and filtered in real-time for various mobile health applications. For instance, for mobile epileptic seizure prediction systems, electroencephalogram (EEG) recordings of brain signals from a 20-channel headset generate about 37 Mbytes per hour for a sample size of 2 bytes and a sampling frequency of 256 Hz [75].

This work proposes a new method for digital filter design that minimizes the computational complexity and the latency of filtering by clustering the filter coefficients so that the frequency response specifications are met. Low latency is achieved by reducing the filter’s processing time in order to realize real-time operation [76]. The computational complexity is measured in terms of the number of multiplications and additions, where a reduction of the number of multiplications shall improve the speed and reduce the power consumption.

Finite impulse response (FIR) filters are desirable since they are stable, have

no limit cycles and have linear phase for (anti-)symmetric coefficients [77]. However, they require more coefficients to achieve a given filter response characteristic, which increases the computational complexity. Reducing the FIR length decreases the complexity but degrades the performance. Hence, the objective of this paper is to design FIR filters that have low computational complexity and maintain the desired frequency response.

Different methods are proposed for FIR filter design. For instance, the frequency sampling approach [78], [79], [77] specifies the desired frequency response at a set of equally spaced frequencies, which is then converted to the FIR filter unit sample response. On the other hand, windowing methods [77], [80] specify the frequency response and then truncate the corresponding unit sample response using windows of Hamming, Kaiser, etc. Another method is the Parks-McClellan (PM) approach which finds the optimal filter coefficients by minimizing the maximum absolute value of the error function between the desired and actual filter response [81], [82], [83], [84], [85]. Evolutionary optimization methods have been introduced recently in [86], [87], [88], [89] and are characterized by their ability to provide global optimization. Moreover, techniques for sparse filter design are proposed in [1], [90], [91] and aim to find FIR filters with sparse coefficients as zero-valued coefficients eliminate multiplications and reduce the complexity.

## 5.2 Proposed Approach

Consider an FIR filter of (anti-)symmetric coefficients  $\{h(n), 0 \leq n \leq N - 1\}$ . Given an input  $\{x(n)\}$ , the output of the filter is given by the convolution of  $\{h(n)\}$  and  $\{x(n)\}$ :

$$y(n) = \sum_{l=0}^{N/2-1} h(l)(x(n-l) + x(n-(N-1-l))) \quad (5.1)$$

Equation (5.1) implies each output sample  $y(n)$  is computed by  $N/2$  multiplications and  $N - 1$  additions, where the  $N/2$  multiplications in (5.1) is achieved by exploiting the fact that filter coefficients  $\{h(l), 0 \leq l \leq N - 1\}$  hold only  $N/2$  distinct values. We now generalize this observation. Consider an even symmetric FIR filter whose coefficients attain  $C$  distinct values  $\{\mu_0, \dots, \mu_{C-1}\}$ ,  $1 \leq C \leq N/2$ . In particular, defining  $\mathcal{I}_k$  as the subset of indices  $\{0, \dots, N/2 - 1\}$  such that  $\forall l \in \mathcal{I}_k, h(l) = h(N - 1 - l) = \mu_k$ , equation (5.1) becomes

$$y(n) = \sum_{k=0}^{C-1} \mu_k \sum_{l=0, l \in \mathcal{I}_k}^{N/2-1} (x(n-l) + x(n-(N-1-l))) \quad (5.2)$$

Equation (5.2) implies that each output sample  $y(n)$  can be computed by  $C$  multiplications and  $N - 1$  additions. In the special case where all coefficients

$\{h(0), \dots, h(N/2 - 1)\}$  are distinct,  $C = N/2$  and (5.2) becomes (5.1). Otherwise,  $C < N/2$  and the number of multiplications to compute an output sample  $y(n)$  in (5.2) is reduced by a factor of  $C/(N/2)$  compared to (5.1). The above observation illustrates that the computational complexity of FIR filtering depends on the number of distinct values within the filter coefficients. Consider an even symmetric FIR filter of coefficients  $\{h(n)\}$ . We seek to find the best approximation  $\{h_c(n)\}$  of  $\{h(n)\}$  such that  $\{h_c(n)\}$  is of same length  $N$ , maintains even symmetry and has at most  $C$  distinct values,  $1 \leq C \leq N/2$ . Denote by  $\{e(n)\}$  the error signal of the output of filters  $\{h(n)\}$  and  $\{h_c(n)\}$  for some input signal  $\{x(n)\}$ , we can write

$$e(n) = \sum_{k=0}^{N-1} (h(k) - h_c(k))x(n-k) \quad (5.3)$$

We define the best approximation  $\{h_c(n)\}$  of  $\{h(n)\}$  as the one that minimizes the energy of the error signal  $\{e(n)\}$  for all input energy signals  $\{x(n)\}$ :

$$\min_{\{h_c(n)\}} \sum_{n=-\infty}^{\infty} |e(n)|^2 \equiv \min_{\{h_c(n)\}} \int_{-\pi}^{\pi} |E(e^{j\omega})|^2 d\omega \quad (5.4)$$

where  $\{h_c(n)\}$  is the set of  $\{h_c(0), \dots, h_c(N-1)\}$  and  $E(e^{j\omega})$  is the frequency response of  $e(n)$ . Let  $H(e^{j\omega})$ ,  $H_c(e^{j\omega})$  and  $X(e^{j\omega})$  be the frequency responses of  $\{h(n)\}$ ,  $\{h_c(n)\}$  and  $\{x(n)\}$  respectively. Incorporating (5.3) into (5.4) and applying the convolution theorem, (5.4) becomes

$$\begin{aligned} & \min_{\{h_c(0), \dots, h_c(N-1)\}} \int_{-\pi}^{\pi} |H(e^{j\omega}) - H_c(e^{j\omega})|^2 |X(e^{j\omega})|^2 d\omega \\ &= \int_{-\pi}^{\pi} \left| \sum_{n=0}^{N-1} (h(n) - h_c(n)) e^{-j\omega n} \right|^2 |X(e^{j\omega})|^2 d\omega \\ &\leq N \left( \sum_{n=0}^{N-1} (h(n) - h_c(n))^2 \right) \int_{-\pi}^{\pi} |X(e^{j\omega})|^2 d\omega \end{aligned} \quad (5.5)$$

The last line of (5.5) follows from the Cauchy-Schwarz inequality and the fact  $|e^{-j\omega n}| = 1 \forall n$ . Equation (5.5) implies that the energy of the error signal  $\{e(n)\}$  of the outputs of the two filters  $\{h(n)\}$  and  $\{h_c(n)\}$  can be suppressed by minimizing the sum square difference of the coefficients of the two filters:

$$\min_{\{h_c(0), \dots, h_c(N-1)\}} \sum_{n=0}^{N-1} (h(n) - h_c(n))^2 \quad (5.6)$$

### 5.2.1 Even Symmetry of Minimizer $\{h_c^*(0), \dots, h_c^*(N-1)\}$

The optimization problem in (5.6) admits the following constraints:

- Optimization variables  $\{h_c(n)\}$  can attain at most  $C$  distinct values,  $1 \leq C \leq N/2$ .
- Variables  $\{h_c(n)\}$  are even symmetric so that minimizer  $\{h_c^*(n)\}$  has a linear phase response

$$h_c(n) = h_c(N - 1 - n), \quad 0 \leq n \leq N - 1 \quad (5.7)$$

We now show that the optimization problem in (5.6) can be relaxed by dismissing the equality constraints in (5.7) without compromising the even symmetry of minimizer  $\{h_c^*(n)\}$ . To prove this, we first define  $\{h_c^{**}(0), \dots, h_c^{**}(N/2 - 1)\}$  as

$$\{h_c^{**}(n)\}_{n=0}^{N/2-1} = \arg \min_{h_c(0), \dots, h_c(N/2-1)} \sum_{n=0}^{N/2-1} (h(n) - h_c(n))^2 \quad (5.8)$$

By the even symmetry of  $\{h(n)\}$ , we also have

$$\{h_c^{**}(n)\}_{n=0}^{N/2-1} = \arg \min_{\{h_c(n)\}_{N/2}} \sum_{n=0}^{N/2-1} (h(N - 1 - n) - h_c(n))^2 \quad (5.9)$$

where  $\{h_c(n)\}_{N/2}$  is the set of  $\{h_c(0), \dots, h_c(N/2 - 1)\}$ . We plug the minimizer  $\{h_c^*(n)\}$  into the objective function (5.6):

$$\begin{aligned} & \sum_{n=0}^{N/2-1} (h(n) - h_c^*(n))^2 + \sum_{n=N/2}^{N-1} (h(n) - h_c^*(n))^2 \\ & \stackrel{a}{\geq} \sum_{n=0}^{N/2-1} (h(n) - h_c^{**}(n))^2 + \sum_{n=0}^{N/2-1} (h(N - 1 - n) - h_c^{**}(n))^2 \quad (5.10) \\ & \stackrel{b}{\geq} \sum_{n=0}^{N-1} (h(n) - h_c^*(n))^2 \end{aligned}$$

where (a) and (b) follow from a change of variables, (5.8)- (5.9) and the definition of  $\{h_c^*(n)\}$ . Since the objective function in (5.6) is strongly convex, minimizer  $\{h_c^*(n)\}$  is unique. Therefore, by (5.10)

$$\begin{aligned} h_c^*(n) &= h_c^{**}(n), \quad 0 \leq n \leq N/2 - 1 \\ h_c^*(n) &= h_c^{**}(N - 1 - n), \quad N/2 \leq n \leq N - 1 \end{aligned} \quad (5.11)$$

which implies  $\{h_c^*(n)\}$  is even symmetric. This also implies that the equality constraints on  $\{h_c(n)\}$  in (5.6) may be dropped, so (5.6) becomes a problem of clustering the original filter coefficients  $\{h(n)\}$  into at most  $C$  distinct values.

### 5.2.2 K-means Based Clustering Solution

The optimization problem in (5.6) can now be solved iteratively using Lloyd's K-means algorithm [92], [93], [94]:

- Select randomly  $C$  filter coefficients of set  $\{h(n), 0 \leq n \leq N - 1\}$  and denote them by  $\{\mu_0, \dots, \mu_{C-1}\}$ .
- Construct set  $\mathcal{I}_k = \{n | 0 \leq n \leq N - 1, |h(n) - \mu_k| \leq |h(n) - \mu_{k'}| \forall k' \neq k\}$  for  $0 \leq k \leq C - 1$ .
- Reinitialize  $\mu_k = \frac{1}{|\mathcal{I}_k|} \sum_{n \in \mathcal{I}_k} h(n)$  for  $0 \leq k \leq C - 1$ , where  $|\mathcal{I}_k|$  is the cardinality of set  $\mathcal{I}_k$ .
- Repeat the last two steps until coefficients' convergence.
- The clustered FIR filter solution of (5.6) is given by

$$h_c^*(n) = \sum_{k=0}^{C-1} \mu_k \times \mathbf{1}\{n \in \mathcal{I}_k\}, \quad 0 \leq n \leq N - 1 \quad (5.12)$$

where  $\mathbf{1}\{\cdot\}$  is the indicator function.

### 5.2.3 Generalized Clustered FIR Filter Approximation

Equation (5.2) shows that the number of multiplications in an FIR filtering operation increases linearly with the number of distinct values of the filter coefficients, or equivalently the number of centroids  $\{\mu_0, \dots, \mu_{C-1}\}$ . On the other hand, (5.6) shows that the sum square difference between filter  $\{h(n)\}$  and its approximation  $\{h_c(n)\}$  decreases with the number of clusters, where it drops from  $(N - 1) \times \text{var}(\{h(n)\})$  for  $C = 1$  to zero for  $C = N/2$ . Therefore, there is a trade-off in selecting the number of clusters between the computational complexity of the filtering operation and the quality of the filter approximation, where the latter should always meet the frequency response specifications. We now show how to improve the filter approximation at a minimal increase in the computational complexity. Note that (5.12) has the form

$$h_c^*(n) = \sum_{k=0}^{C-1} b_{n,k} \mu_k, \quad 0 \leq n \leq N - 1 \quad (5.13)$$

i.e., each coefficient  $h_c^*(n)$  in the clustered filter approximation is a weighted sum of the centroids  $\{\mu_0, \dots, \mu_{C-1}\}$  solution of (5.6). Given the centroids, our objective is to optimize the selection of weights  $\{b_{n,k}\}$  in (5.13) so that  $\{h_c^*(n)\}$  better



approximates  $\{h(n)\}$  and maintains the filtering operation computationally efficient. Plugging (5.13) into (5.1) we have

$$y_c(n) = \sum_{k=0}^{C-1} \mu_k \sum_{l=0}^{N/2-1} b_{l,k} [x(n-l) + x(n-(N-1-l))] \quad (5.14)$$

Note that weights  $\{b_{n,k}\}$  in (5.13) become multiplicative terms in the filtering operation of (5.14). We therefore impose the constraints

$$b_{n,k} \in \{0\} \cup \{\pm 2^z \mid z \in \mathcal{Z}\} \quad \forall n, k \quad (5.15)$$

where  $\mathcal{Z}$  is the set of integers. Since multiplication by zero or unity is trivial and multiplication by powers of two is a bitwise-shift  $\mathcal{O}(1)$  operation, by (5.15) the number of non-trivial multiplications in (5.14) is  $C$  as in (5.2). However, while in (5.2) the number of additions is  $N-1$ , in (5.14) this becomes

$$N-1 + \sum_{l=0}^{N/2-1} \left[ \left( \sum_{k=0}^{C-1} \mathbf{1}\{b_{l,k} \neq 0\} \right) - 1 \right] \quad (5.16)$$

We therefore impose an additional constraint on weights  $\{b_{n,k}\}$  in (5.13), namely that filter coefficient  $h_c^*(n)$  can be mapped to at most two cluster centroids

$$\left( \sum_{k=0}^{C-1} \mathbf{1}\{b_{l,k} \neq 0\} \right) \in \{1, 2\} \quad \forall l \quad (5.17)$$

Equations (5.16) and (5.17) imply that the number of additions in (5.14) is in the range  $[N-1, 3N/2-1]$ . Since additions are much cheaper than multiplications, the computational complexity of the filtering operation in (5.14) is therefore comparable to that in (5.2) under constraints (5.15) and (5.17).

The optimal selection of  $\{b_{n,k}\}$  in (5.13) seeks to minimize the sum square difference between  $\{h(n)\}$  and  $\{h_c^*(n)\}$ . Algorithm 1 shows how to optimally map each filter coefficient  $h_c^*(n)$  to either the closest centroid or the midpoint of the two closest centroids. Note the similarity between (5.18) and (5.12) which shows that general mapping (5.13) of  $h_c^*(n)$  to  $\{\mu_k\}$  virtually increases the number of centroids from  $C$  to  $2C-1$  to better approximate filter  $\{h(n)\}$ . Algorithm 1 can be easily extended to the case where  $\{b_{n,k}\}$  are general powers of two as in (5.15). Algorithm 1 can also be extended to the case where coefficients  $\{h(n)\}$  are complex numbers.

---

**Algorithm 1**

---

**Input:**  $\{h(0), \dots, h(N-1)\}$  and  $C$ **Output:**  $\{h_c^*(0), \dots, h_c^*(N-1)\}$ 

- 1: Solve (5.6) for centroids  $\{\mu_0, \dots, \mu_{C-1}\}$  as in Section 5.2.2.
- 2: Sort  $\{\mu_0, \dots, \mu_{C-1}\}$  into  $\{\mu_{i_0}, \dots, \mu_{i_{C-1}}\}$
- 3: Let  $m_k = (\mu_{i_k} + \mu_{i_{k+1}})/2$ ,  $0 \leq k \leq C-2$ . Form set  $\{\hat{\mu}_0, \dots, \hat{\mu}_{2C-2}\} = \{\mu_{i_k}\} \cup \{m_k\}$ .
- 4: Construct set  $\hat{\mathcal{I}}_k = \{n | 0 \leq n \leq N-1, |h(n) - \hat{\mu}_k| \leq |h(n) - \hat{\mu}_{k'}| \forall k' \neq k\}$  for  $0 \leq k \leq 2C-2$ .
- 5: Return  $\{h_c^*(n)\}$  where

$$h_c^*(n) = \sum_{k=0}^{2C-2} \hat{\mu}_k \times \mathbf{1}\{n \in \hat{\mathcal{I}}_k\}, \quad 0 \leq n \leq N-1 \quad (5.18)$$

---

### 5.3 Simulation Results and Performance Analysis

Figure 5.1 presents the magnitude response over normalized frequency for lowpass (LPF) and highpass (HPF) filters designed using the proposed (with  $C=20$ ), Hamming window and Parks-McClellan (PM) methods. The LPFs have passband cutoff frequency  $w_p = 0.25$ , stopband cutoff frequency  $w_s = 0.3$  and filter length  $N = 72$ , while HPFs have  $w_s = 0.25$ ,  $w_p = 0.3$  and  $N = 72$ . We make two observations:

- Although the approximation filters are designed for the same set of specifications ( $w_p$ ,  $w_s$ ,  $N$ ,  $C$ ), they have a different frequency response. This is because the clustering method seeks to meet a set of specifications only indirectly by minimizing as in (5.4) the energy of the error signal of two filter outputs, that of an original filter  $\{h(n)\}$  meeting the specifications and an approximate filter  $\{h_c(n)\}$ . Thus, for different  $\{h(n)\}$ , a different solution for  $\{h_c(n)\}$  is generated.
- The magnitude of the approximation of the PM filter closely matches that of the original filter both in the passband and the stopband. This is only true in the passband in the case of the Hamming window, where the gap in the stopband grows for higher frequencies.

The second observation can be understood by inspecting the frequency response of the HPF in Figure 5.1. As in the case of the LPF, a close match is observed in the PM case. However, this is now also true in the Hamming window case for the magnitude response in both the passband and the stopband. The justification for

Table 5.1: Performance of PM, sparse, proposed and ideal filters in the stopband and passband. LPF ( $\omega_p = 0.5, \omega_s = 0.55$ ), HPF ( $\omega_p = 0.55, \omega_s = 0.5$ ),  $N = 62$  coefficients,  $C=20$  clusters (proposed) and  $Z=42$  zeros (sparse [1]).

		$MSE_s$	$MSE_p$	$\delta_{s_{max}}$ (dB)	$\delta_{p-p}$ (dB)
LPF	Ideal	0	0	$-\infty$	0
	PM	0.0919	0.0927	-77.1889	0.8396
	Sparse	0.5514	0.1149	-50.5638	1.6131
	Proposed	0.0924	0.0964	-69.3012	0.9921
HPF	Ideal	0	0	$-\infty$	0
	PM	0.0919	0.0930	-77.2520	0.8416
	Sparse	0.55	0.1120	-51.1201	1.6240
	Proposed	0.1161	0.0893	-73.4245	1.0390

the clustering method performance in the Hamming window case in Figure 5.1 is that a Hamming window has a smooth differentiable form unlike a PM filter. Thus, the smoothness form of the Hamming window is lost upon clustering, which adds to the high frequency content of the filter magnitude response. In the case of a lowpass Hamming window filter, the energy content of the filter response is concentrated in the low frequency band, so the high frequency *loss-of-smoothness* noise is non-negligible. For a highpass Hamming window filter, the energy content is concentrated in the high frequency band and thus this noise becomes negligible.

In Figure 5.2a, the proposed method is compared against the original PM filter for different number of clusters  $C$ . The figure shows that the clustering method can be computationally very efficient as the response changes notably only for very small  $C$  values (e.g.,  $C = 5$ ). This also implies that the clustering method yields a reduction in the number of multiplications for a single filtering operation by a factor of  $(C)/(N/2) = C/36$  (for  $N = 72$ ). For  $C = 10$ , over 70% computational savings can be achieved. This amount of reduction makes the designed filter well-suited for low-end devices and for processing of big-data signals.

We have also applied the proposed filter design to a large EEG dataset from the Freiburg database [95], which includes around 200 GB of non-invasive data for 21 patients. The signals were sampled at 256 Hz, with 16 bits per sample. Using the proposed method with  $C = 13$  clusters, the complexity is estimated to be around 97 million multiplications per hour, which represents a significant reduction when compared with 600 million multiplications per hour when using the Parks McClellan filter (with  $N=72$  coefficients for both filters).

Figure 5.2b shows the magnitude frequency response using the proposed method based on an even symmetric PM filter ( $\omega_p = 0.5, \omega_s = 0.55, N = 62$ ) for number of distinct filter coefficients  $C_1 = 20$  and  $C_2 = 13$ . Figure 5.2b shows the magnitude response of two non-symmetric filters designed using the sparse

filter of [1]. The number of zero-valued coefficients ( $Z_1 = N - C_1 = 42$  and  $Z_2 = N - C_2 = 49$ ) is selected such that each filter using the proposed method and the corresponding sparse filter have the same number of multiplications. Figure 5.2b shows that the proposed method provides a tighter transition band and more equiripple stopband response compared to the sparse filter method for the high computational complexity case. In addition, the proposed method maintains the equiripple response and achieves sharper cutoff in the transition band and higher attenuation in the stopband for the low computational complexity case. Moreover, the response using the sparse filter method significantly varies for the two complexities compared to the proposed method. This shows that the proposed method performs better compared to state-of-the-art methods for the same computational saving.

Table 5.1 presents additional performance metrics for the various filter design methods including the mean squared errors  $MSE_s$ ,  $MSE_p$  for the stopband and passband, respectively, the maximum peak in the stopband ( $\delta_{smax}$ ), and the peak-to-peak deviation in the passband ( $\delta_{p-p}$ ). The MSEs are computed with respect to an ideal filter.

## 5.4 Conclusion

In this part of the thesis, we presented a new digital filter design characterized by low complexity, suitable for low-power and big data signal processing applications. The design method incorporates K-means clustering and a mapping technique of the filter coefficients. Simulation results show that the designed filter has flat response in the passband, a narrow transition band, and high attenuation in the stopband. The method also offers a better performance-computational complexity trade-off compared to state-of-the-art digital filter design methods.

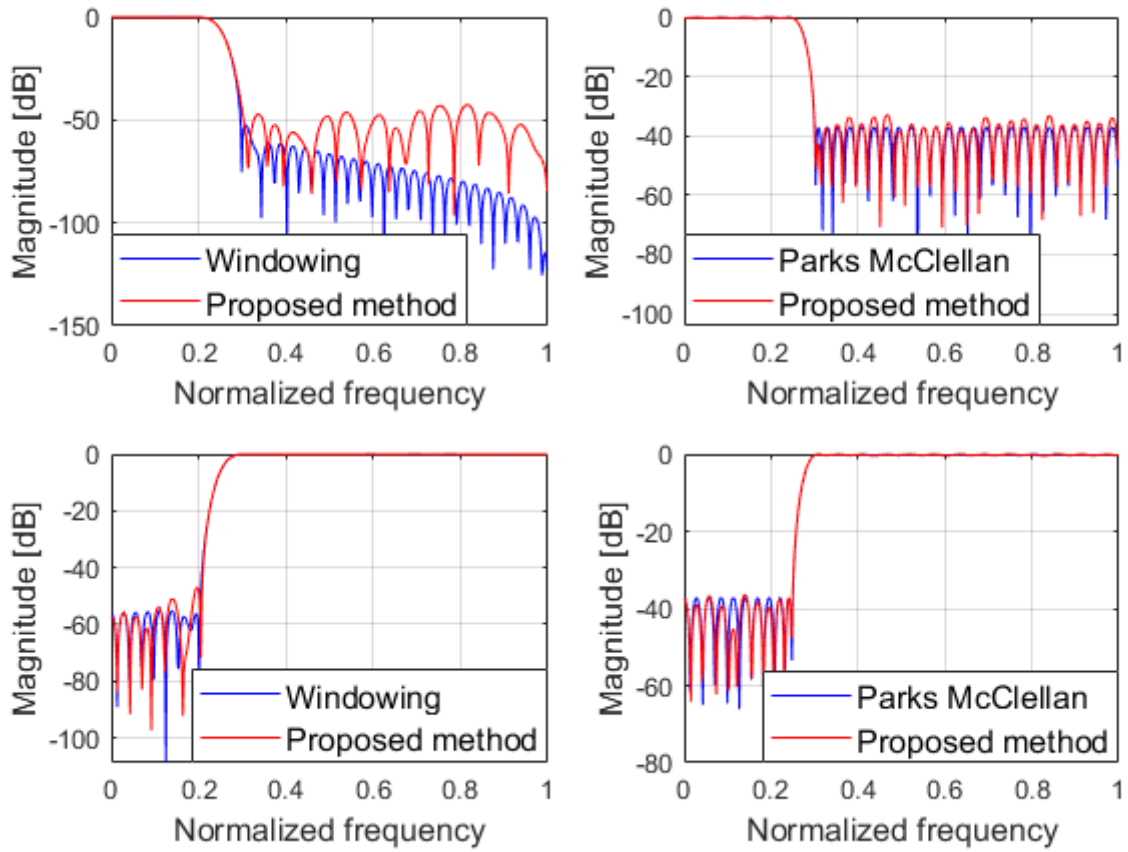


Figure 5.1: Comparison of the frequency response of LPF ( $w_p = 0.25$ ,  $w_s = 0.3$ ,  $N = 72$ ) and HPF ( $w_p = 0.3$ ,  $w_s = 0.25$ ,  $N = 72$ ) using Hamming window, Parks-McClellan and the proposed method with  $C = 20$ .

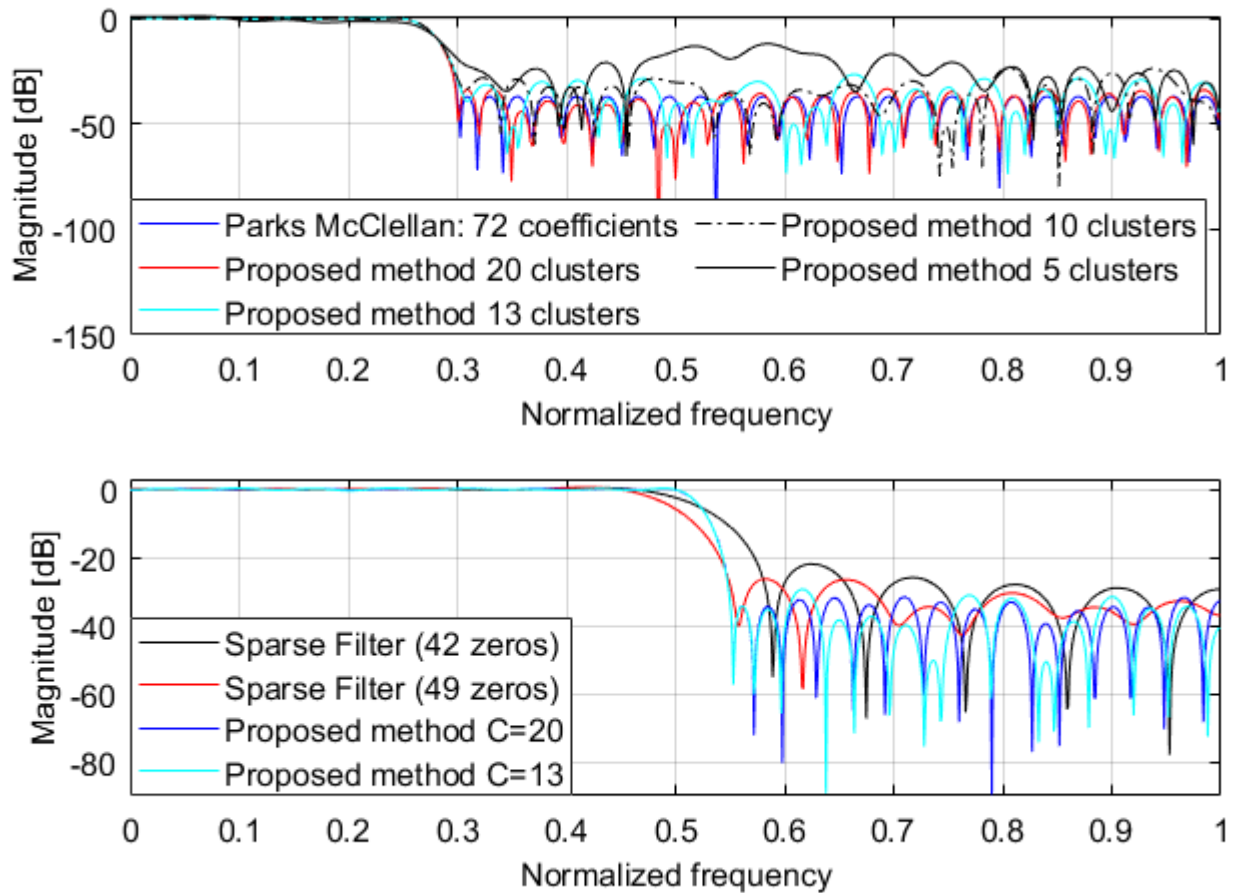


Figure 5.2: (a) Effect of the number of clusters on the passband and stopband gains. (b) Comparison between the proposed method and the sparse filter method of [1] with  $N = 62$ ,  $\omega_p = 0.5$  and  $\omega_s = 0.55$ .

## Chapter 6

# Dynamic Compression for Big Biomedical Data

This is the final part of the thesis, where we present our work in the compression of electroencephalogram (EEG) signals. The development of a neurologically-oriented mobile health system involves significant challenges in terms of the proper sensing and efficient transmission of EEG signals, and the faithful reconstruction of these signals at the receiving node. EEG compression has been widely used to reduce storage requirements, improve the real time processing of the sensed signals, and provide a better and timely feedback to the concerned patients. The non-stationarity of the EEG signals and the large volumes of data being continuously processed mandate the development of data reduction schemes that provide a good trade-off between compression performance and the preservation of the signal quality and integrity. To this end, we propose in this part of the thesis a dynamic and effective compression approach for EEG data that relies on a sequence of compression and decompression phases to optimize the compression rate while maintaining a distortion level below a target threshold. Simulation results using real EEG data segments show that even with stringent quality requirements, a notable compression ratio can be attained with minimal processing overhead.

### 6.1 Background and Literature Review

The technological developments in the design of wearable EEG headsets has permitted the ambulatory monitoring of brain disorders [96]. In particular, the advancements in the design of scalp electrodes has facilitated the accurate and continuous sensing of the brain's electrical activity while the development of miniaturized capable, and frugal signal processing hardware has contributed to the ubiquitous and quasi-real time processing of the sensed signal. This has paved the way to the establishment of neurologically-oriented mobile health solutions

aiming at the continuous management of brain disorders which involves two main steps: the detection and prediction of the disorder and the feedback that alarms the user of the occurrence of an abnormal brain activity to prevent its debilitating effects.

The continuous monitoring process involves the processing and transmission of large volumes of data. This creates a bottleneck as the efficiency of the health solution relies on how fast the data is being stored, transmitted and processed. For instance, an hour of EEG recordings from a 20-channel headset generates around 37 Mbytes/hour (2 bytes/sample, 256 Hz sampling frequency) of EEG data [97]. In order to reduce the impact of the collected data on the system performance, data reduction techniques have been investigated to try to send significantly less data while keeping the key characteristics of the EEG signals upon recovery at the receiver side. Data reduction is further divided into two separate tracks, namely, data reduction through selective transmission and data reduction through compression and feature extraction. Selective transmission employs special algorithms to decide which sensed data to transmit and when. On the other hand, data reduction aims at reducing the volume of data being transmitted. In this context, the compression of EEG signals offers notable prospects into achieving this goal, by removing redundancies (lossless compression), or even dropping some signal information (lossy compression).

Compression schemes have been extensively studied for the reduction of EEG signals. Due to the importance of preserving data integrity, several lossless algorithms were presented such as as Lempel-Ziv coding, Huffman coding were used for EEG data and provided some notable compression performance [98]. It is also argued in [98] that employing predictors increases the compression ratio. Examples include Markov predictors, linear predictive coding (LPC) and artificial neural network (ANN)-based predictors. Lossy compression schemes were also recently studied in order to reduce the volumes of bio-electric data being processed and transmitted. Many algorithms in the literature build on the set partitioning in hierarchical trees (SPIHT) image-based compression algorithm [99] such as in [100] where redundancy between EEG frequency subbands is exploited and used to develop a SPIHT-based compression algorithm. Moreover, a low-complexity lossy compression scheme using a variant of the SPIHT algorithm is presented in [101]. In [102], independent component analysis, and a medical compression technique known as Waves, are used to develop a lossy algorithm that provides notable data reduction. Compressed sensing was recently used for EEG data reduction [103], [104]. In [104], a technique known as the block sparse Bayesian learning (BSBL) is used in conjunction with the sensing technique to address the low sparsity of EEG signals.

Due to the importance of maintaining the integrity of biomedical data, several metrics were used to measure the loss of signal fidelity between the original signal and the reconstructed one. Examples include the percentage root-mean-square difference (PRD), the root mean square error and the cross correlation.



Increasing the compression ratio while maintaining such parameters at an acceptable level is a notable challenge taking into account the non-stationarity of EEG signals. This property makes the performance of a compression algorithm dependent on the EEG segment being processed. To address this challenge, we propose a dynamic learning-based compression scheme which tries, iteratively, to achieve a high compression ratio while maintaining an appropriate PRD level. For a given EEG segment, this takes place by dynamically tuning the percentage of segments being processed using lossless and lossy compression, for a limited number of iterations in order to achieve a target PRD level.

## 6.2 Proposed EEG Compression Algorithm

The proposed dynamic compression algorithm starts from some predetermined compression parameters (training phase) and adjusts them iteratively for each EEG segment being processed, to reach a minimum PRD while preserving a high compression ratio (dynamic adjustment phase). The training and dynamic adjustment phases involve running a lossy compression routine several times, each time with a different set of initial parameters.

The compression routine presented in Figure 6.1 was initially proposed in our previous work in [105]. It uses a combination of discrete cosine transform (DCT) and adaptive differential pulse coded modulation (ADPCM). The DCT takes correlated input data and concentrates its energy in just the first few transform coefficients. This allows for aggressive compression to take place by treating each subset of the coefficients differently. The first few coefficients are preserved through lossless compression while the remainder of the coefficients are either subject to lossy compression or discarded. The resulting vector of DCT coefficients is divided into four portions. The main aim is to match the decreasing importance of the coefficients by an increasing level of lossiness in the compression. Thus, the first  $\alpha\%$  of the coefficients block is subject to a lossless compression operation. The next  $\beta\%$  of the block is subject to ADPCM with a 16:4 compression ratio. Then,  $\gamma\%$  is subject to a more aggressive lossy compression with a 10:2 compression ratio. The remaining part of the block is discarded as this contains the least relevant coefficients for signal reconstruction.

The complementary operations are performed in order to decompress the compressed EEG signal. These operations include lossless decoding, ADPCM decoding and inverse DCT (IDCT). The quality of the reconstructed signal is assessed by calculating the PRD metric which quantifies the distortion in the received signal and the fidelity in the signal reconstruction. The PRD metric is given as

$$\text{PRD} = \sqrt{\frac{\sum_{n=1}^N (x(n) - \hat{x}(n))^2}{\sum_{n=1}^N x(n)^2}} \quad (1)$$

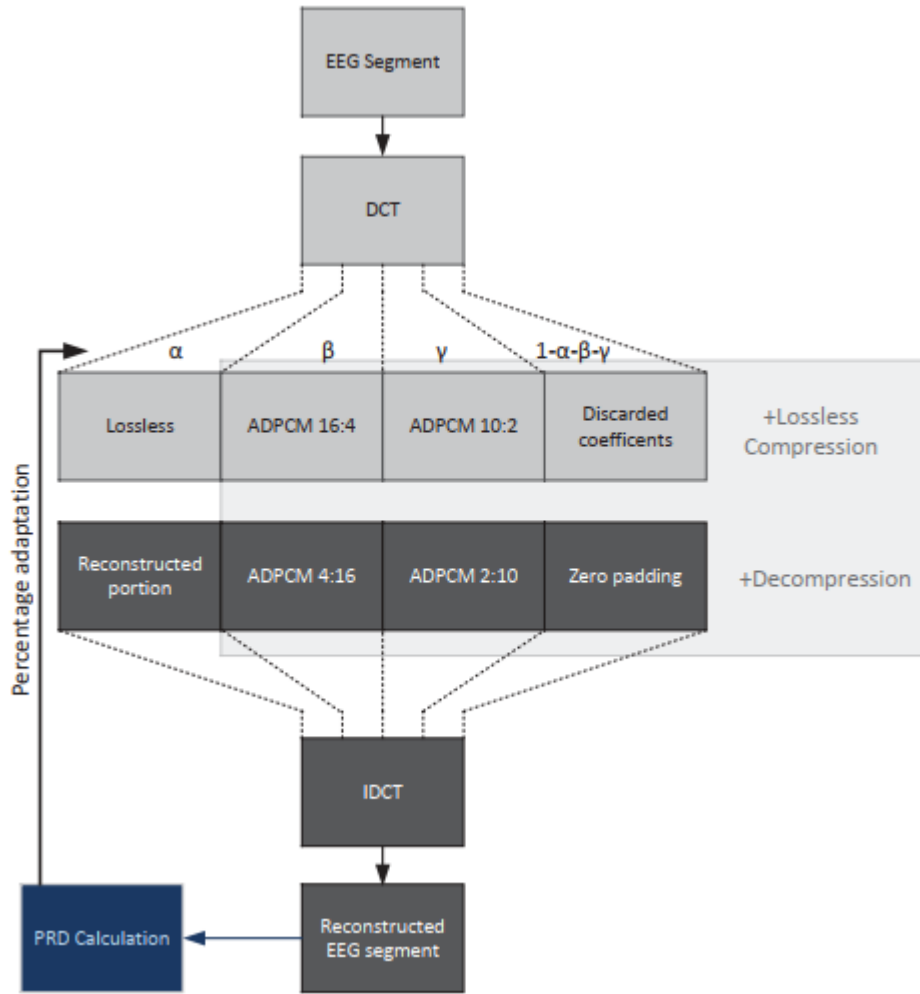


Figure 6.1: Compression routine used in the training and dynamic adjustment phases.

where  $x(n)$  and  $\hat{x}(n)$  are the original and reconstructed signals, respectively.  $n$  is the sample index and  $N$  is the total number of samples in the original and reconstructed EEG segments.

### 6.2.1 Training Phase

In the training phase, an exhaustive search is performed using different combinations of the  $\alpha$ ,  $\beta$  and  $\gamma$  values. Each parameter is divided into a discrete set of eleven values (0% to 100%). The combinations are selected such that the proportions sum to 100%. The combinations were tested for a target PRD of 30% and 7% and the combination yielding the highest compression ratio for each target

PRD and given in (2), is selected as the most suitable compression strategy. This is also considered as the reference strategy, which would be refined through the proposed dynamic compression technique.

$$\text{CR} = \frac{\text{original bits} - \text{compressed bits}}{\text{original bits}} \times 100 \quad (2)$$

## 6.2.2 Dynamic Parameter Adjustment Phase

The main purpose of this approach is to try and improve the PRD (by reducing it) for a given EEG window while maintaining a high compression ratio. An intuitive approach would be to increase the percentage of lossless compression considerably (increase  $\alpha$ ). This however has a big impact on the compression ratio which would decrease considerably. Alternatively, the increments in the values of  $\alpha$  and  $\beta$  will be done in an iterative manner in order to reach a lower target PRD (TPRD) without having a drastic impact on the compression ratio. The starting values for this dynamic approach are selected from the reference values determined in the training phase as described in Section 6.2.1. Then the algorithm highlighted in Figure 6.2 operates as follows

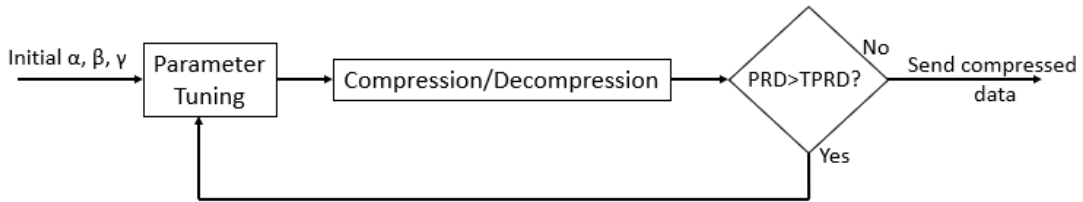


Figure 6.2: Dynamic compression approach.

- **Parameter Tuning** The parameters are tuned gradually to reduce the PRD value starting by solely decreasing the level of lossy compression (increasing  $\beta$ , decreasing  $\gamma$  and reducing the set of discarded coefficients) and then gradually increasing the lossless compression. To this end, important parameters need to be set such as the percentage of increase/decrease in the different portions and the number of times tuning takes place. In this work, the iterative parameter tuning goes as follows. While the PRD is greater than TPRD, the chain will be executed at most four times. In the first iteration,  $\gamma$  is increased by 15% thus reducing the aggressively compressed portions ( $\gamma$  and discarded data). In the second iteration,  $\alpha$  is increased by 5% while decreasing  $\gamma$  and the set of discarded coefficients. In the third iteration, both  $\alpha$  and  $\beta$  are increased by 5% and 15%, respectively. Finally, in the last iteration,  $\alpha$  is increased by 5% which are now deduced from the  $\beta$  portion. The rationale behind this selection is to try and avoid lossless

data compression while relying on the less aggressive 16:4 ADPCM compression to reduce the PRD value. This also justifies the selection of a 5% increase for the lossless compression segment compared to 15% for the 16:4 ADPCM part. Thus, the algorithm will try to reduce the aggressiveness in the lossy compression before relying on increasing the lossless compression percentage to get the PRD value below the TPRD.

- **Compression/Decompression Iterations** The compression and decompression are performed as described in Figure 6.1. The decompression is done at each iteration in order to be able to assess the performance of the compression resulting from the parameter tuning in terms of the achieved PRD.

- **PRD Calculation** The PRD value is calculated as in (1). An illustration of the operation of the algorithm is shown in Figure 3 for patient 1 of the Freiburg database (<http://epilepsy-database.eu/>). After training, the initial parameters are set to  $\alpha = 0\%$ ,  $\beta = 30\%$ ,  $\gamma = 10\%$  and the remaining part is discarded. The initial PRD is 47.3% whereas the initial compression ratio is 94.1%. After applying the proposed algorithm, the PRD dropped to 10% while the compression ratio reached 92% after the fourth iteration. This trend was also observed for other patients as no more than four iterations were needed to achieve the required PRD threshold.

### 6.2.3 Complexity Analysis and Practical Considerations

The proposed compression approach is mainly designed to operate in the context of a mobile health system, thus requiring a relatively low computational complexity. Among the different constituting blocks of the compression and decompression algorithms (Lossless, ADPCM, DCT/IDCT), the DCT algorithm has the largest complexity requirement of  $n \log n$  multiplications where  $n$  is the number of samples being processed by the DCT algorithm in each window (e.g., one minute window is 15360 samples assuming a sampling rate of 256 Hz). The DCT coefficients are calculated once per window, and the compression using the different  $\alpha$ ,  $\beta$ ,  $\gamma$  proportions are applied directly to the DCT coefficients. On the other hand, IDCT is computed at each iteration as part of the decompression. This is done at most four times since a maximum of four iterations can be used. Therefore, the complexity of the DCT/IDCT computations sums up to  $5n \log n$  multiplications which is considered fast even for the miniaturized signal processing hardware on the wearable headsets.

Patient ID	162	239	261	891
Initial CR	94.15	94.3	94.2	94.33
Optimized CR	93	92	92	93
Initial PRD	78.52	57.96	57.1	70.97
Optimized PRD	9.9	8.35	9.42	9.56
Stopping iteration	2	3	2	3

Table 6.1: Sample testing outcome with a TPRD of 10%.

## 6.3 Performance Evaluation

The proposed compression approach was tested on real EEG data obtained from the Freiburg database. For the training phase in the proposed algorithm, 60 one-minute EEG segments (windows) from a 20-channel setup are selected, thus resulting in 1200 EEG segments. Different combinations were tested to determine the best  $\alpha, \beta$  and  $\gamma$  values that yield the best compression ration for a target PRD value.

### 6.3.1 EEG Compression Results

In order to emulate the compression process in a realistic scenario, and to ensure independence between the training and testing data sets, separate blocks of data were selected for training and testing. For each of the patients, testing was performed on several one-hour EEG blocks which were divided into one-minute segments. The proposed approach was applied to these segments and the average compression ratio and PRD values calculated. Table 6.1 shows the average compression ratio and PRD for four patients data from the Freiburg database. For each patient, an initial  $\alpha = 0\%$  indicating no lossless compression and a  $\beta = 30\%$  indicating a processing of 30% of the EEG segment using 16:4 ADPCM, were selected. For the same starting point, the PRD values were different from patient to patient as they are not the optimal ones resulting from the training phase. The selection of a common starting point also highlights the need for a dynamic compression scheme where the parameters should be tweaked per patient and for each processed EEG segment. As a general trend, it can be noted that less than four iterations were needed to achieve the target PRD value of 10% for a relatively high starting PRD value. Concurrently, the compression ratio is above 90% for all the cases. These conclusions demonstrate that only a small number of iterations is needed to converge to the targeted value.

**1) Comparison with Other Algorithms** In order to further assess the performance gains of the proposed approach, the results presented in Table 6.1

are compared to the performance of the JPEG2000 and SPIHT algorithms [99]. These two algorithms were selected as they are widely used in the literature for EEG compression( [100], [106]). The JPEG2000 algorithm relies on a sequence of wavelet transform, quantization and arithmetic coding. The SPIHT algorithm, on the other hand, relies on wavelet transform, set partitioning sorting, and entropy coding. To test these algorithms, the sampled EEG signals are reshaped into two-dimensional matrices on a channel by channel basis. These matrices are then processed using the compression algorithms and average results are reported.

Figure 6.3 illustrates the resulting compression ratio for each of the four patients in Table 6.1. For fair comparison, the parameters were tweaked for the other two approaches to achieve a PRD value below the target one. As shown in Figure 6.3, the proposed approach outperforms the other two methods for all the patients. This is mainly due to the dynamic compression feature of our approach. At each iteration and for each EEG window, the compression parameters are being tuned to ensure the highest compression ratio while keeping the PRD below the target value. The figure also shows that SPIHT achieves the closest compression ratios. However, this comes at the expense of increased computational complexity due to two inherent features, namely, the use of arithmetic coding and the use of floating-point operations which require longer data length. One additional factor with an impact on complexity comes from calculating wavelet transform that depends on the length of the wavelet filters, which is at least one multiplication per coefficient. As for the JPEG2000 algorithm, the computational complexity is not significant; however, the compression performance is well below the proposed algorithm.

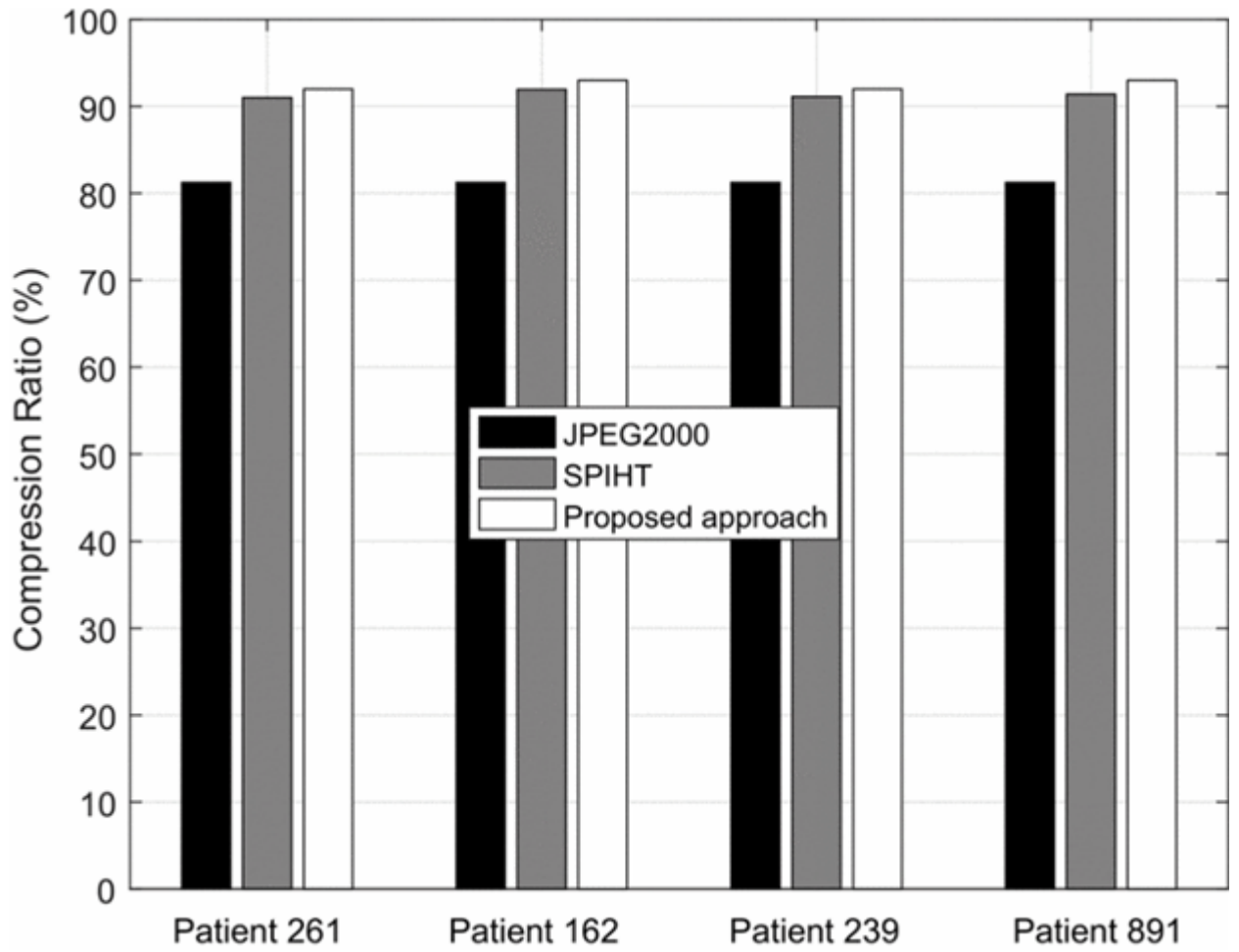


Figure 6.3: Comparison of the proposed algorithm, SPIHT and JPEG2000 with a TPRD of 10%.

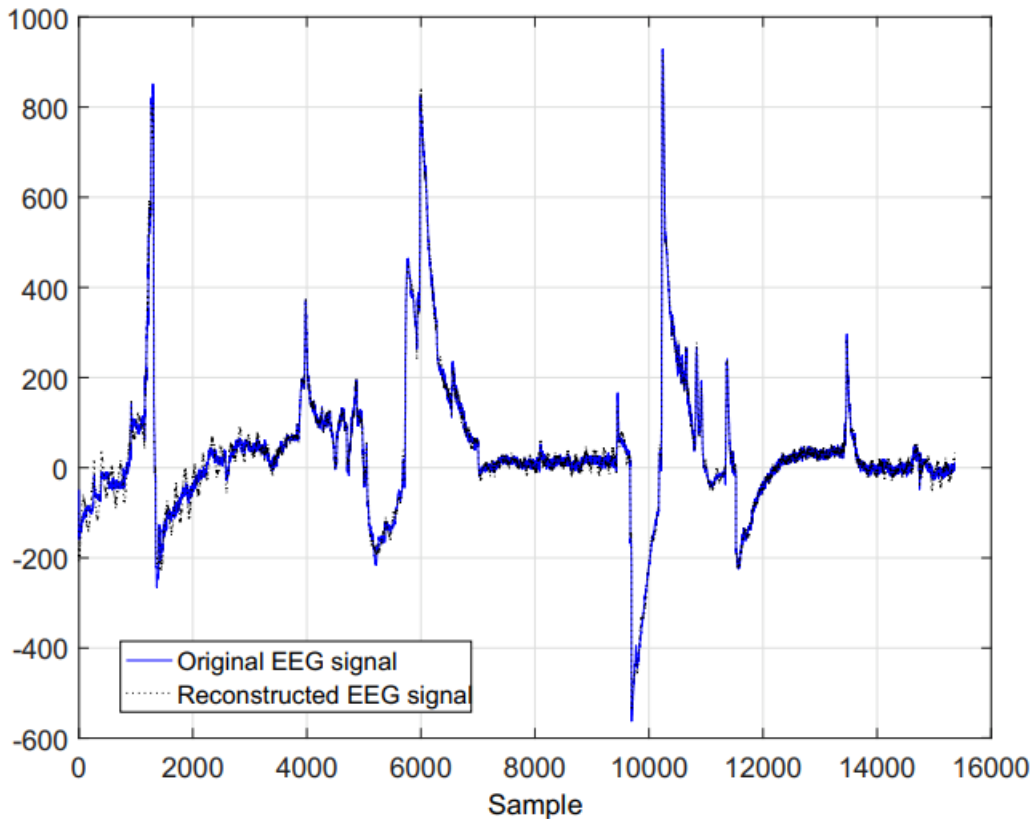


Figure 6.4: Original and reconstructed EEG segment.

### 6.3.2 Faithfulness in Signal Reconstruction

As the main goal of a mobile health system is to provide continuous monitoring of the patient’s medical condition, the correct reconstruction of the EEG signal, after transmission to the processing device, is very important. To measure the fidelity in the reconstruction, we consider two aspects of EEG analysis, namely, visual and feature based analysis. The first one is used by neurologists to depict certain events that reflect the patient’s condition. The second is mainly used to come up with automated detection and prediction techniques of neurological disorders.

Figure 6.4 shows an example EEG segment and the resulting output after the compression/decompression and reconstruction process. It can be noticed that the reconstructed signal is nearly identical to the original one. The reconstruction process preserved the signal variations including the spikes. Therefore, the reconstructed signal can be used by specialists to delimit neurological events such as a seizure occurrence.



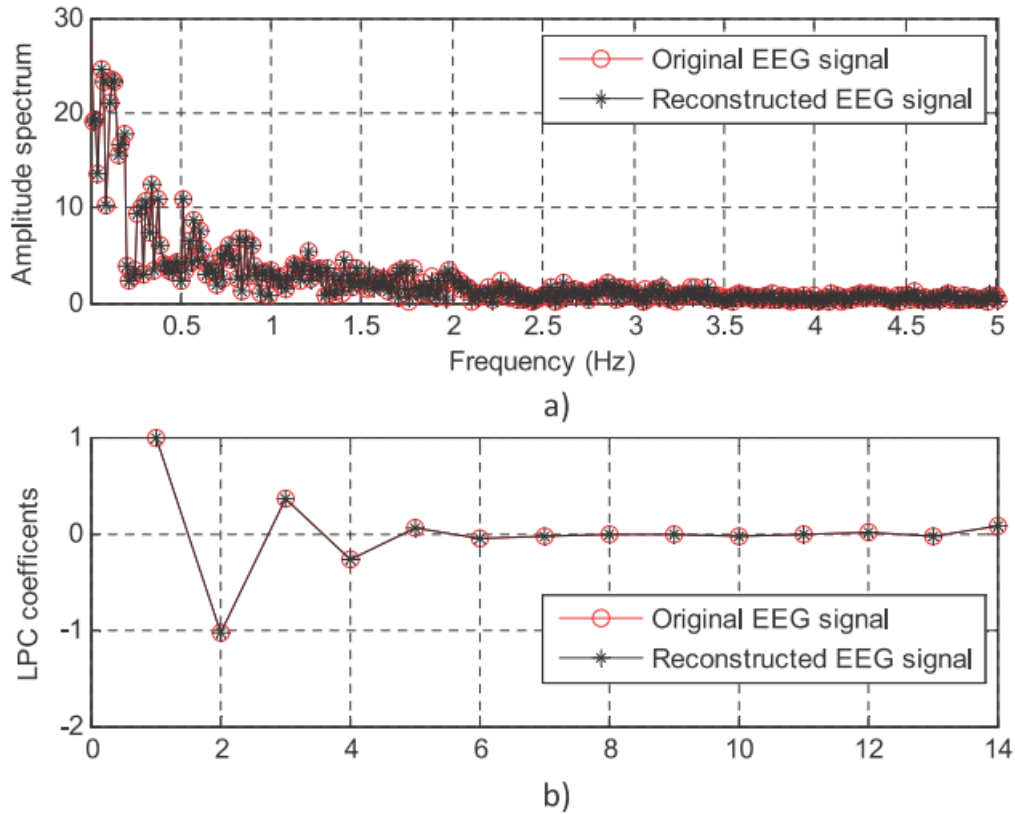


Figure 6.5: Fourier transform coefficients (a) and linear predictive coding (LPC) coefficients of the original and reconstructed signals.

Figure 6.5 presents further evidence on the quality of the reconstruction through a comparison of the coefficients obtained through two feature extraction techniques, the Fourier transform and linear predictive coding. We can see that the signal's features are preserved by the proposed compression algorithm. Thus, instead of working directly on the sensed signal, the compressed version can be transmitted to the mobile device for further processing. This has a significant impact in terms of energy saving as a smaller amount of data needs to be stored, transmitted and processed.

### 6.3.3 Complexity

In Section 6.2.3, the computational complexity of the proposed algorithm was analyzed. We observed that the most complex operation is in the DCT/IDCT. Since a neurologically oriented mobile health system involves contiguous and continuous multi-channel measurements which need to be processed and analyzed, it is important to check how the algorithm scales when larger EEG segments are being processed. To investigate this issue, the algorithm was applied to EEG

segments of different window sizes. The results shown in Figure 6.6 provide an idea of the running time of the proposed compression algorithm. The processing time for a one minute segment is around 0.71 seconds with an overhead of 1.18%. For a ten minutes segment, the compression approach takes about 6.5 seconds with a similar overhead. Thus, the proposed approach incurs a delay which is very small when compared to the size of the EEG segment being processed.

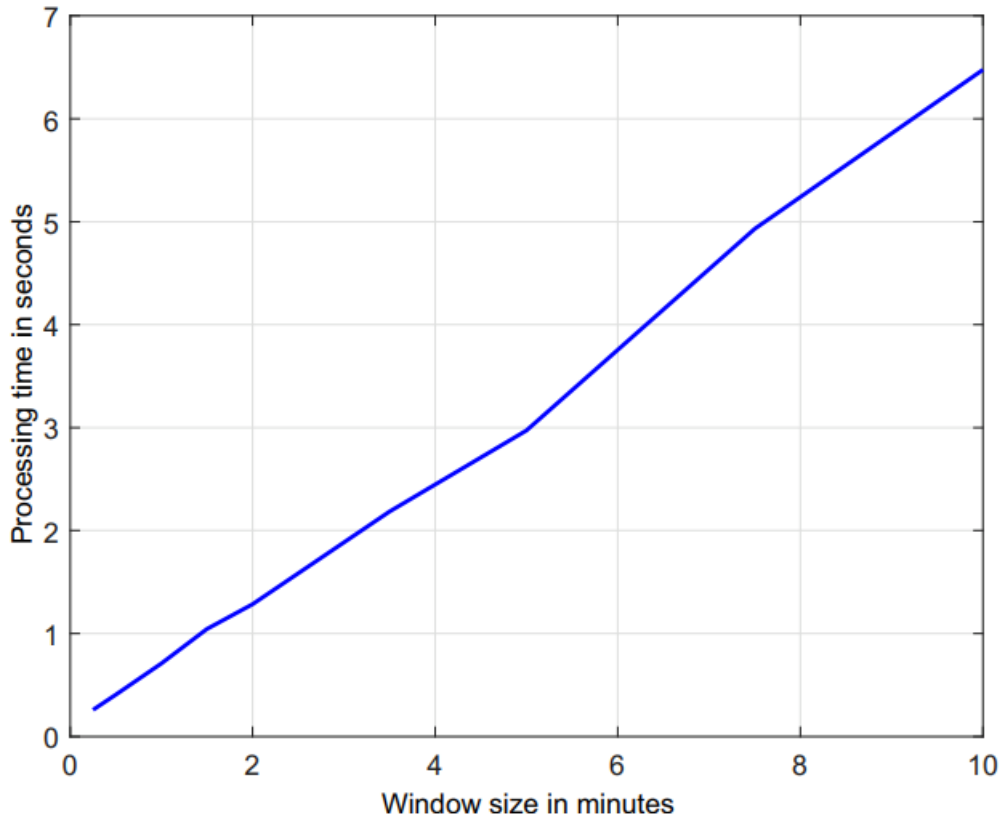


Figure 6.6: Running time of proposed algorithm for different EEG segment sizes.

## 6.4 Conclusion

In this chapter, a dynamic EEG compression algorithm was proposed. The scheme applies a sequence of compression/decompression operations in order to find the best loss-less/lossy compression combination that provides the highest compression ratio for each EEG segment while preserving the signal integrity. Performance evaluation results on real EEG datasets demonstrate the gains achieved through the proposed scheme in terms of high compression ratio, low percentage root-mean-square difference and low computational overhead.

# Chapter 7

## Future Research Directions

Many challenging topics have been studied in this thesis including the design of a distributed learning algorithm among agents with heterogeneous computational capabilities, approximation of the gradient descent algorithm, design of low complexity FIR filters and dynamic compression approach for big EEG data. The findings of this thesis revealed further research directions and raised new open questions, which can drive future research efforts and boost even more the interest in these areas. Below are some suggested future research directions that can be investigated as an extension to the contributions of the thesis.

- Optimizing the performance of distributed learning over heterogeneous networks.
  - Optimal edge weighting: We mean here to investigate a new method which can find the optimal weights that can be assigned to the shared parameters over the network by the capable and the non-capable agents. Intuitively speaking, the incoming parameters from the capable agents should be given more weight than the incoming ones from the non-capable agents. To find the optimal weights, this can be set as an optimization problem, which looks for the optimal weights over the edges of the heterogeneous network that can minimize the excess risk function of the network.
  - Dynamic operation among optimization algorithms: We mean here to propose a dynamic strategy that allows the distributed agents to alter among different optimization algorithms based on some power consumption restrictions. For instance, some agents may be able to run Newton's or gradient descent methods, but due to limited energy resources at some duration, these agents may decide to shift to low complexity algorithms like the mini-batch or the stochastic gradient descent algorithms. The switching from one algorithm to another during offline/online learning leads to reserving the available power, and

its impact on the mean square deviation and the excess risk should be studied.

- Learning over agents with heterogeneous computational capabilities under non-convex functions: We mean here to study the scenario when the distributed agents own non-convex cost functions and different computational capabilities. An important application of this study is when the distributed agents are running the neural networks. Training neural networks is very challenging and may take hours or days of training. The best general optimization algorithm known for training neural networks is the stochastic gradient descent (SGD), where model weights are updated at each iteration using the back-propagation algorithm. Letting non-capable agents cooperate with agents who can run faster algorithms than SGD (like mini-batch GD) will end up in reducing the training time over distributed neural networks.
- Heterogeneous learning over weakly connected networks: We mean here to study the impact of distributed learning among agents with heterogeneous computational capabilities over weakly connected networks. Distributed learning over weakly connected networks arise in important situations. For example, in the context of interactions over social networks where weak connectivity can be used to model the ability of authoritarian governments to manage the flow of media information, or to model the behavior of stubborn agents who never update their opinions and may represent political parties, media sources or leaders, trying to influence the beliefs in the rest of the society.
- A clustering-based approach for designing sparse FIR filters: This is another important research direction that can be investigated to design a low complexity FIR filter such that the coefficients are sparse and have at most at most a given set of distinct values. The sparsity of the coefficients and the upper bound over the number of distinct values can be set as constraints for an optimization problem. The importance of this work is in providing very low complexity FIR filters that are suitable for low-power and big data signal processing applications.

# Chapter 8

## Thesis Publications

- Hybrid Distributed Optimization for Learning over Networks with Heterogeneous Agents, submitted as a journal paper and it is under review.
- A Low Complexity Approximation of Gradient Descent For Learning Over Single And Multi-Agent Systems, submitted as a conference paper and it is under review.
- A Clustering-Based Approach for Designing Low Complexity FIR Filters [107], published in IEEE Signal Processing Letter, 2021.
- Dynamic EEG Compression Approach with Optimized Distortion Level for Mobile Health Solutions [108], published in IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom), Munich, 2016.
- On EEG Lossy Data Compression for Data-Intensive Neurological Mobile Health Solutions [105], published in International Conference on Advances in Biomedical Engineering (ICABME), Beirut, 2015.

# Appendices

# Appendix A

## A.1 Network Mean-Square-Error Stability

Multiplying both sides of (3.16) by  $\mathcal{V}_\epsilon^T$  we get

$$\mathcal{V}_\epsilon^T \tilde{\omega}_i = \mathcal{V}_\epsilon^T \mathcal{B}_{i-1} \tilde{\omega}_{i-1} + \mathcal{V}_\epsilon^T \mathcal{M} s_i(\omega_{i-1}) - \mathcal{V}_\epsilon^T \mathcal{M} b \quad (\text{A.1})$$

$$\begin{bmatrix} \bar{\omega}_i \\ \check{\omega}_i \end{bmatrix} = \begin{bmatrix} I - D_{11,i-1}^T & -D_{21,i-1}^T \\ -D_{12,i-1}^T & \mathcal{J}_\epsilon^T - D_{22,i-1}^T \end{bmatrix} \begin{bmatrix} \bar{\omega}_{i-1} \\ \check{\omega}_{i-1} \end{bmatrix} + \begin{bmatrix} \bar{s}_{i-1} \\ \check{s}_{i-1} \end{bmatrix} - \begin{bmatrix} \bar{b} \\ \check{b} \end{bmatrix} \quad (\text{A.2})$$

where

$$\mathcal{V}_\epsilon^T \tilde{\omega}_i = \begin{pmatrix} (p^T \otimes I_M) \tilde{\omega}_i \\ (V_R^T \otimes I_M) \tilde{\omega}_i \end{pmatrix} = \begin{pmatrix} \bar{\omega}_i \\ \check{\omega}_i \end{pmatrix} \quad (\text{A.3})$$

$$\mathcal{V}_\epsilon^T \mathcal{M} s(\psi_{i-1}) = \begin{pmatrix} (p^T \otimes I_M) \mathcal{M} s(\psi_{i-1}) \\ (V_R^T \otimes I_M) \mathcal{M} s(\psi_{i-1}) \end{pmatrix} = \begin{pmatrix} \bar{s}_i \\ \check{s}_i \end{pmatrix} \quad (\text{A.4})$$

$$\mathcal{V}_\epsilon^T \mathcal{M} b = \begin{pmatrix} (p^T \otimes I_M) \mathcal{M} b \\ (V_R^T \otimes I_M) \mathcal{M} b \end{pmatrix} = \begin{pmatrix} \bar{b} \\ \check{b} \end{pmatrix} \quad (\text{A.5})$$

Note that, from the expression of  $\bar{b}$  and  $\check{b}$ , we note that they depend on  $\mathcal{M}$  and  $b$ . Recall from (3.24) that the entries of  $b$  are defined in terms of the gradient vectors and  $\nabla J_k(\omega^o)$  and the matrix  $Y_{k,o}^\delta = Y_{k,i-1}^\delta(\omega^o)$ . Since each  $J_k(\omega)$  is assumed to be twice-differentiable, then each  $\nabla J_k(\omega)$  is a differentiable function and therefore bounded. In addition,  $Y_{k,i-1}^\delta$  is also bounded as discussed in Section 3.2.5. It follows that

$$\bar{b} = - \sum_{k=1}^N p_k \mu_k Y_{k,o}^\delta \nabla J_k(\omega^o) = O(\mu_{max}) \quad (\text{A.6})$$

Similarly,

$$\check{b} = O(\mu_{max}) \quad (\text{A.7})$$

From (A.2) we can write

$$\bar{\omega}_i = (I - D_{11,i-1}^T) \bar{\omega}_{i-1} - (D_{21,i-1}^T) \check{\omega}_{i-1} + \bar{s}_{i-1} \quad (\text{A.8})$$

$$\check{\omega}_i = (-D_{12,i-1}^T)\bar{\omega}_{i-1} + (\mathcal{J}_\epsilon^T - D_{22,i-1}^T)\check{\omega}_{i-1} + \check{s}_{i-1} - \check{b} \quad (\text{A.9})$$

Conditioning both sides of (A.8) on  $\mathcal{F}_{i-1}$  and using the condition on gradient noise that is given in (3.45),

$$\begin{aligned} \mathbf{E}[\|\bar{\omega}_i\|^2 | \mathcal{F}_{i-1}] &= \mathbf{E}[\|(I - D_{11,i-1}^T)\bar{\omega}_{i-1} - (D_{21,i-1}^T)\check{\omega}_{i-1} + \bar{s}_{i-1} - \bar{b}\|^2 | \mathcal{F}_{i-1}] \\ &= \|(I - D_{11,i-1}^T)\bar{\omega}_{i-1} - (D_{21,i-1}^T)\check{\omega}_{i-1} - \bar{b}\|^2 + \mathbf{E}[\|\bar{s}_{i-1}\|^2 | \mathcal{F}_{i-1}] \end{aligned} \quad (\text{A.10})$$

Conditioning again on both sides of (A.10) we get

$$\begin{aligned} \mathbf{E}[\|\bar{\omega}_i\|^2] &= \mathbf{E}[\|(I - D_{11,i-1}^T)\bar{\omega}_{i-1} - (D_{21,i-1}^T)\check{\omega}_{i-1} - \bar{b}\|^2] + \mathbf{E}[\|\bar{s}_{i-1}\|^2] \\ &\leq \frac{1}{1-t} \mathbf{E}[\|(I - D_{11,i-1}^T)\bar{\omega}_{i-1}\|^2] + \frac{1}{t} \mathbf{E}[\|(D_{21,i-1}^T)\check{\omega}_{i-1} - \bar{b}\|^2] + \mathbf{E}[\|\bar{s}_{i-1}\|^2] \\ &\stackrel{a}{\leq} \frac{1}{1-t} \mathbf{E}[\|(I - D_{11,i-1}^T)\|^2 \|\bar{\omega}_{i-1}\|^2] + \frac{2}{t} \mathbf{E}[\|(D_{21,i-1}^T)\|^2 \|\check{\omega}_{i-1}\|^2] + \frac{2}{t} \|\bar{b}\|^2 + \mathbf{E}[\|\bar{s}_{i-1}\|^2] \\ &\stackrel{b}{=} \frac{(1 - \sigma_{11}\mu_{max})^2}{1 - \sigma_{11}\mu_{max}} \mathbf{E}[\|\bar{\omega}_{i-1}\|^2] + \frac{2\sigma_{21}^2\mu_{max}^2}{\sigma_{11}\mu_{max}} \mathbf{E}[\|\check{\omega}_{i-1}\|^2] + \frac{2\|\bar{b}\|^2}{\sigma_{11}\mu_{max}} + \mathbf{E}[\|\bar{s}_{i-1}\|^2] \\ &= (1 - O(\mu_{max})) \mathbf{E}[\|\bar{\omega}_{i-1}\|^2] + O(\mu_{max}) \mathbf{E}[\|\check{\omega}_{i-1}\|^2] + O(\mu_{max}) + \mathbf{E}[\|\bar{s}_{i-1}\|^2] \end{aligned} \quad (\text{A.11})$$

It is shown in [2] that  $\mathbf{E}[\|\bar{s}_{i-1}\|^2]$  and  $\mathbf{E}[\|\check{s}_{i-1}\|^2]$  are upper bounded by

$$\begin{aligned} &\mathbf{E}[\|\bar{s}_{i-1}\|^2], \mathbf{E}[\|\check{s}_{i-1}\|^2] \\ &\leq \nu_1^2 \nu_2^2 \beta_d^2 \mu_{max}^2 [\mathbf{E}[\|\bar{\omega}_i\|^2] + \mathbf{E}[\|\check{\omega}_i\|^2]] + \nu_1^2 \mu_{max}^2 \sigma_s^2 \end{aligned} \quad (\text{A.12})$$

Substituting (A.12) in (A.11) we get,

$$\mathbf{E}[\|\bar{\omega}_i\|^2] \leq (1 - O(\mu_{max})) \mathbf{E}[\|\bar{\omega}_{i-1}\|^2] + O(\mu_{max}) \mathbf{E}[\|\check{\omega}_{i-1}\|^2] + O(\mu_{max}^2) \quad (\text{A.13})$$

Similarly, we get

$$\mathbf{E}[\|\check{\omega}_i\|^2] \leq (\rho(J_\epsilon) + \epsilon + O(\mu_{max}^2)) \mathbf{E}[\|\check{\omega}_{i-1}\|^2] + O(\mu_{max}^2) \mathbf{E}[\|\bar{\omega}_{i-1}\|^2] + O(\mu_{max}^2) \quad (\text{A.14})$$

Equations (A.13) and (A.14) can be written in a matrix form as follows

$$\begin{aligned} \begin{bmatrix} \mathbf{E}[\|\bar{\omega}_i\|^2] \\ \mathbf{E}[\|\check{\omega}_i\|^2] \end{bmatrix} &\leq \Gamma \begin{bmatrix} \mathbf{E}[\|\bar{\omega}_{i-1}\|^2] \\ \mathbf{E}[\|\check{\omega}_{i-1}\|^2] \end{bmatrix} + \begin{bmatrix} O(\mu_{max}^2) \\ O(\mu_{max}^2) \end{bmatrix} \\ &= \Gamma \begin{bmatrix} \mathbf{E}[\|\bar{\omega}_{i-1}\|^2] \\ \mathbf{E}[\|\check{\omega}_{i-1}\|^2] \end{bmatrix} + \begin{bmatrix} O(\mu_{max}^2) \\ O(\mu_{max}^2) \end{bmatrix} \end{aligned} \quad (\text{A.15})$$

where  $\Gamma$  is a matrix that is given by

$$\Gamma = \begin{bmatrix} 1 - O(\mu_{max}) & O(\mu_{max}^2) \\ O(\mu_{max}^2) & \rho(J_\epsilon) + \epsilon + O(\mu_{max}^2) \end{bmatrix} \quad (\text{A.16})$$



Since  $\rho(J_\epsilon) < 1$  is independent of  $\mu_{max}$ , and since  $\epsilon$  and  $\mu_{max}$  are small positive numbers that can be chosen arbitrarily small and independently of each other, then it can be easily shown that

$$\begin{aligned} \limsup_{i \rightarrow \infty} \begin{bmatrix} \mathbf{E} \|\bar{\omega}_i\|^2 \\ \mathbf{E} \|\check{\omega}_i\|^2 \end{bmatrix} &\leq (I - \Gamma)^{-1} \begin{bmatrix} O(\mu_{max}^2) \\ O(\mu_{max}^2) \end{bmatrix} \\ &= \begin{bmatrix} O(\frac{1}{\mu_{max}}) & O(1) \\ O(\mu_{max}) & O(1) \end{bmatrix} \begin{bmatrix} O(\mu_{max}^2) \\ O(\mu_{max}^2) \end{bmatrix} = \begin{bmatrix} O(\mu_{max}) \\ O(\mu_{max}^2) \end{bmatrix} \end{aligned} \quad (\text{A.17})$$

from which we conclude that

$$\limsup_{i \rightarrow \infty} \mathbf{E} \|\bar{\omega}_i\|^2 = O(\mu_{max}) \quad (\text{A.18})$$

$$\limsup_{i \rightarrow \infty} \mathbf{E} \|\check{\omega}_i\|^2 = O(\mu_{max}^2) \quad (\text{A.19})$$

and therefore

$$\begin{aligned} \limsup_{i \rightarrow \infty} \mathbf{E} \|\tilde{\omega}_i\|^2 &= \limsup_{i \rightarrow \infty} \mathbf{E} \|(\mathcal{V}_\epsilon^{-1})^T \begin{bmatrix} \bar{\omega}_i \\ \check{\omega}_i \end{bmatrix}\|^2 \\ &\leq \limsup_{i \rightarrow \infty} \|(\mathcal{V}_\epsilon^{-1})^T\|^2 [\mathbf{E} \|\bar{\omega}_i\|^2 + \mathbf{E} \|\check{\omega}_i\|^2] = O(\mu_{max}). \end{aligned} \quad (\text{A.20})$$

■

## A.2 Network Fourth-Order Moment Stability

Using (A.8), and (A.14), we can write

$$\|\bar{\omega}_i\|^4 = \|(I - D_{11,i-1}^T)\bar{\omega}_{i-1} - (D_{21,i-1}^T)\check{\omega}_{i-1} + \bar{s}_{i-1}\|^4 \quad (\text{A.21})$$

$$\|\check{\omega}_i\|^4 = \|-D_{12,i-1}^T\bar{\omega}_{i-1} + (\mathcal{J}_\epsilon^T - D_{22,i-1}^T)\check{\omega}_{i-1} + \check{s}_{i-1} - \check{b}\|^4 \quad (\text{A.22})$$

Using the following inequality

$$\|a + b\|^4 \leq \|a\|^4 + 3\|b\|^4 + 8\|a\|^2\|b\|^2 + 4\|a\|^2\text{Re}(a * b) \quad (\text{A.23})$$

Let

$$a = (I - D_{11,i-1}^T)\bar{\omega}_{i-1} - (D_{21,i-1}^T)\check{\omega}_{i-1} \quad (\text{A.24})$$

$$b = \bar{s}_{i-1} \quad (\text{A.25})$$

then decompose the right-hand side of (A.21) according to (A.23), and take the expectations on both sides and follow exactly the same steps in [2], we get

$$\limsup_{i \rightarrow \infty} \mathbf{E} \|\bar{\omega}_i\|^4 = O(\mu_{max}^2) \quad (\text{A.26})$$

Similarly, let

$$a = (-D_{12,i-1}^T)\bar{\omega}_{i-1} + (\mathcal{J}_\epsilon^T - D_{22,i-1}^T)\check{\omega}_{i-1} - \check{b} \quad (\text{A.27})$$

$$b = \check{s}_{i-1} \quad (\text{A.28})$$

and decompose the right-hand side of (A.22) according to (A.23), and take the expectations on both sides and follow exactly the same steps in [2], we get

$$\limsup_{i \rightarrow \infty} \mathbf{E} \|\check{\omega}_i\|^4 = O(\mu_{max}^4) \quad (\text{A.29})$$

and, therefore

$$\begin{aligned} \limsup_{i \rightarrow \infty} \mathbf{E} \|\tilde{\omega}_i\|^4 &= \limsup_{i \rightarrow \infty} \mathbf{E} \left( \left\| (\mathcal{V}_\epsilon^{-1})^T \begin{bmatrix} \bar{\omega}_i \\ \check{\omega}_i \end{bmatrix} \right\|^2 \right)^2 \\ &\leq \limsup_{i \rightarrow \infty} \left\| (\mathcal{V}_\epsilon^{-1})^T \right\|^4 [\mathbf{E} \|\bar{\omega}_i\|^2 + \mathbf{E} \|\check{\omega}_i\|^2] = O(\mu_{max}^2). \end{aligned} \quad (\text{A.30})$$

■

### A.3 Network Mean-Error Stability

$$\tilde{\omega}_i = \mathcal{B}_{i-1} \tilde{\omega}_{i-1} + \mathcal{M}s(\psi_{i-1}) - \mathcal{M}b, i \geq 0 \quad (\text{A.31})$$

$$\mathcal{B}_{i-1} = (\mathcal{I} - \mathcal{M}\mathcal{H}'_{i-1})\mathcal{A}_1^T \quad (\text{A.32})$$

Let

$$\tilde{\mathcal{H}} = \mathcal{H} - \mathcal{H}'_{i-1} \quad (\text{A.33})$$

where  $\mathcal{H}$  is a constant matrix defined as

$$\mathcal{I} = \text{diag}\{I_1, I_2, \dots, I_N\} \quad (\text{A.34})$$

and  $I_k, k = 1, \dots, N$  represents the eye matrix.

Substituting (A.33) in (A.32), we can write

$$\mathcal{B}_{i-1} = \mathcal{B} + \mathcal{M}\tilde{\mathcal{H}}\mathcal{A}_1^T \quad (\text{A.35})$$

where  $\mathcal{B}$  is given by

$$\mathcal{B} = (I_{MN} - \mathcal{M}\mathcal{H})\mathcal{A}_1^T \quad (\text{A.36})$$

Substituting (A.35) in (A.31), we get

$$\tilde{\omega}_i = \mathcal{B}\tilde{\omega}_{i-1} + \mathcal{M}\tilde{\mathcal{H}}\mathcal{A}_1^T\tilde{\omega}_{i-1} + \mathcal{M}s(\psi_{i-1}) - \mathcal{M}b, i \geq 0 \quad (\text{A.37})$$

Taking conditional expectations on both sides of (A.37)

$$\mathbf{E}[\tilde{\omega}_i | \mathcal{F}_{i-1}] \stackrel{a}{=} \mathcal{B}\tilde{\omega}_{i-1} + \mathcal{M}\tilde{\mathcal{H}}\mathcal{A}_1^T\tilde{\omega}_{i-1} - \mathcal{M}b, i \geq 0 \quad (\text{A.38})$$

where (a) follows from  $\mathbf{E}[s(\psi_{i-1})|\mathcal{F}_{i-1}] = 0$ . Taking the expectations of both sides of (A.38), we can write

$$\begin{aligned}\mathbf{E}[\tilde{\omega}_i] &= \mathcal{B}\mathbf{E}[\tilde{\omega}_{i-1}] + \mathcal{M}\mathbf{E}[\tilde{\mathcal{H}}\mathcal{A}_1^T\tilde{\omega}_{i-1}] - \mathcal{M}b, i \geq 0 \\ &= \mathcal{B}\mathbf{E}[\tilde{\omega}_{i-1}] + \mathcal{M}c_{i-1} - \mathcal{M}b, i \geq 0\end{aligned}\quad (\text{A.39})$$

where  $c_{i-1}$  is defined as follows

$$c_{i-1} = \mathbf{E}[\tilde{\mathcal{H}}\mathcal{A}_1^T\tilde{\omega}_{i-1}] \quad (\text{A.40})$$

Multiplying both sides of (A.39) by  $\mathcal{V}_\epsilon^T$ , we can write

$$\begin{bmatrix} \mathbf{E}\bar{\omega}_i \\ \mathbf{E}\tilde{\omega}_i \end{bmatrix} = \bar{\mathcal{B}} \begin{bmatrix} \mathbf{E}\bar{\omega}_{i-1} \\ \mathbf{E}\tilde{\omega}_{i-1} \end{bmatrix} - \begin{bmatrix} 0 \\ \check{b} \end{bmatrix} + \mathcal{V}_\epsilon^T \mathcal{M}c_{i-1} \quad (\text{A.41})$$

Note that (A.41) has the same form as in [2], with the exception that  $c_{i-1}$  and  $\tilde{\mathcal{H}}$  take different forms in this work from that in [2]. So, we present below the upper bounds of  $\tilde{\mathcal{H}}$ , and  $\|\mathcal{V}_\epsilon^T \mathcal{M}c_{i-1}\|$  and then the remaining steps of the proof follow similarly as in [2].

$$\begin{aligned}\tilde{H}_{k,i-1}^\delta &= I - \mathbf{H}_{k,i-1}'^\delta \\ \|\tilde{H}_{k,i-1}^\delta\| &= \|I - \mathbf{H}_{k,i-1}'^\delta\| \stackrel{a}{\leq} L_{max}\mathcal{K}'_dCN\|\tilde{\omega}_{i-1}\|\end{aligned}\quad (\text{A.42})$$

where (a) follows from (3.44), then

$$\|\tilde{\mathcal{H}}_{i-1}\| = \max_{1 \leq k \leq N} \|\tilde{H}_{k,i-1}^\delta\| \leq L_{max}\mathcal{K}'_dCN\|\tilde{\omega}_{i-1}\| \quad (\text{A.43})$$

therefore

$$\begin{aligned}\|\mathcal{V}_\epsilon^T \mathcal{M}c_{i-1}\| &\leq \|\mathcal{V}_\epsilon^T\| \|\mathcal{M}\| \|c_{i-1}\| \\ &\leq \|\mathcal{V}_\epsilon^T\| \|\mathcal{M}\| \|\mathbf{E}(\tilde{\mathcal{H}}_{i-1}\mathcal{A}_1^T\tilde{\omega}_{i-1})\| \\ &\leq (\|\mathcal{V}_\epsilon^T\| \|\mathcal{M}\| \|\mathcal{A}_1^T\|) \mathbf{E}[\|\tilde{\mathcal{H}}_{i-1}\| \|\tilde{\omega}_{i-1}\|] \\ &\stackrel{a}{\leq} L_{max}\mathcal{K}'_dCN\|\mathcal{V}_\epsilon^T\| \|\mathcal{M}\| \|\mathcal{A}_1^T\| \mathbf{E}\|\tilde{\omega}_{i-1}\|^2 \\ &\stackrel{b}{=} r\mu_{max}\mathbf{E}\|\tilde{\omega}_{i-1}\|^2\end{aligned}\quad (\text{A.44})$$

for some constant  $r$  that is independent of  $\mu_{max}$ . It follows from (3.68) that

$$\limsup_{i \rightarrow \infty} \|\mathcal{V}_\epsilon^T \mathcal{M}c_{i-1}\|^2 = O(\mu_{max}^2) \quad (\text{A.45})$$

Rewrite (A.41) as

$$\begin{bmatrix} \bar{z}_i \\ \check{z}_i \end{bmatrix} = \bar{\mathcal{B}} \begin{bmatrix} \bar{z}_{i-1} \\ \check{z}_{i-1} \end{bmatrix} - \begin{bmatrix} 0 \\ \check{b} \end{bmatrix} - \begin{bmatrix} \bar{b}' \\ \check{b}' \end{bmatrix} - \begin{bmatrix} \bar{\mathcal{B}}' \bar{z}_{i-1} \\ \check{\mathcal{B}}' \check{z}_{i-1} \end{bmatrix} + \begin{bmatrix} \bar{c}_{i-1} \\ \check{c}_{i-1} \end{bmatrix} + \begin{bmatrix} \bar{c}'_{i-1} \\ \check{c}'_{i-1} \end{bmatrix} \quad (\text{A.46})$$

then continue as in [2], we get

$$\limsup_{i \rightarrow \infty} \|\mathbf{E}\tilde{\omega}_{k,i}\| = O(\mu_{max}), \text{ for } k = 1, 2, \dots, N. \quad (\text{A.47})$$

## A.4 Derived Results from Lipschitz Condition

By noting that  $\nabla P(\omega^*) = 0$  and using the Lipschitz condition of  $\nabla P(\cdot)$  in (4.9), we have

$$\mathbf{Result\ 1:} \quad \|\nabla P(\omega_{n-1})\| = \|\nabla P(\omega^*) - \nabla P(\omega_{n-1})\| \leq \delta \|\tilde{\omega}_{n-1}\| \quad (\text{A.48})$$

Substituting  $\omega_2 = \omega^*$  and  $\omega_1 = \omega_{n-1}$  in (4.8), we get

$$\nabla P(\omega_{n-1})^T \tilde{\omega}_{n-1} \leq P(\omega^*) - P(\omega_{n-1}) - \frac{\nu}{2} \|\tilde{\omega}_{n-1}\|^2 \quad (\text{A.49})$$

Further substituting  $\omega_2 = \omega_{n-1}$  and  $\omega_1 = \omega^*$  in (4.8), we get

$$P(\omega^*) - P(\omega_{n-1}) \leq -\frac{\nu}{2} \|\tilde{\omega}_{n-1}\|^2 \quad (\text{A.50})$$

(A.50) and (A.49) imply

$$\mathbf{Result\ 2:} \quad \nabla P(\omega_{n-1})^T \tilde{\omega}_{n-1} \leq -\nu \|\tilde{\omega}_{n-1}\|^2 \quad (\text{A.51})$$

## Bibliography

- [1] M. Nakamoto, T. Itani, and K. Konishi, “Optimal Least-Squares Design of Sparse FIR Filters for Big-Data Signal Processing,” in *IEEE 23rd International Conference on Digital Signal Processing (DSP)*, pp. 1–5, Nov 2018.
- [2] A. H. Sayed, *Adaptation, Learning, and Optimization over Networks*. 2014.
- [3] V. Solo, “Stability of distributed adaptive algorithms i: Consensus algorithms,” in *2015 54th IEEE Conference on Decision and Control (CDC)*, pp. 7422–7427, 2015.
- [4] D. Jin, J. Chen, C. Richard, J. Chen, and A. H. Sayed, “Affine Combination of Diffusion Strategies Over Networks,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 2087–2104, 2020.
- [5] A. H. Sayed, “Diffusion Adaptation over Networks,” *CoRR*, vol. abs/1205.4220, 2012.
- [6] J. Lee, S. Kim, W. Song, and A. H. Sayed, “Spatio-Temporal Diffusion Strategies for Estimation and Detection Over Networks,” *IEEE Transactions on Signal Processing*, vol. 60, no. 8, pp. 4017–4034, 2012.
- [7] S. Tu and A. H. Sayed, “Diffusion Strategies Outperform Consensus Strategies for Distributed Estimation Over Adaptive Networks,” *IEEE Transactions on Signal Processing*, vol. 60, no. 12, pp. 6217–6234, 2012.
- [8] J. Chen, C. Richard, and A. H. Sayed, “Multitask Diffusion Adaptation Over Networks With Common Latent Representations,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 3, pp. 563–579, 2017.
- [9] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, “Equation of State Calculations by Fast Computing Machines,” *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.
- [10] Lin Xiao and S. Boyd, “Fast Linear Iterations for Distributed Averaging,” in *42nd IEEE International Conference on Decision and Control (IEEE Cat. No.03CH37475)*, vol. 5, pp. 4997–5002 Vol.5, 2003.

- [11] W. K. Hastings, “Monte Carlo Sampling Methods Using Markov Chains and Their Applications,” *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.
- [12] L. J. Billera and P. Diaconis, “A Geometric Interpretation of the Metropolis-Hastings Algorithm,” *Statist. Sci.*, vol. 16, pp. 335–339, 11 2001.
- [13] A. Sayed, *Adaptive Filters*. Wiley - IEEE, Wiley, 2011.
- [14] B. Bollobás, *Modern Graph Theory*. Graduate Texts in Mathematics 184, Springer-Verlag New York, 1 ed., 1998.
- [15] D. M. Cvetković, M. Doob, and H. Sachs, *Spectra of Graphs: Theory and Application*. New York: Academic Press, 1980.
- [16] W. Kocay and D. L. Kreher, *Graphs, Algorithms and Optimization*. Chapman Hall/CRC, 2004.
- [17] J. Nocedal and S. J. Wright, *Numerical Optimization*. New York, NY, USA: Springer, second ed., 2006.
- [18] I. Griva, S. G. Nash, and A. Sofer, *Linear and Nonlinear Optimization (2. ed.)*. SIAM, 2008.
- [19] K. Yuan, B. Ying, S. Vlaski, and A. H. Sayed, “Stochastic gradient descent with finite samples sizes,” in *IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2016.
- [20] D. Yin, A. Pananjady, and M. Lam, “Gradient diversity: a key ingredient for scalable distributed learning,” in *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2018.
- [21] H. Robbins and S. Monro, “A Stochastic Approximation Method,” *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.
- [22] N. Qian, “On the Momentum Term in Gradient Descent Learning Algorithms,” *The Official Journal of the International Neural Network Society*, vol. 12, no. 1, pp. 145–151, 1999.
- [23] Y. Nesterov, “A Method for Unconstrained Convex Minimization Problem with the Rate of Convergence  $O(1/k^2)$ ,” *Doklady ANSSSR (translated as Soviet.Math.Docl.)*, vol. 269, pp. 543–547, 1983.
- [24] H. E. . S. Y. Duchi, J., “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization,” *Machine Learning Research*, vol. 12, pp. 2121–2159, 2011.
- [25] M. D. Zeiler, “ADADELTA: An Adaptive Learning Rate Method,” *Retrieved from <http://arxiv.org/abs/1212.5701>*, 2012.

- [26] . B. J. L. Kingma, D. P., “Adam: a Method for Stochastic Optimization,” in *International Conference on Learning Representations*, pp. 1–13, 2015.
- [27] T. Dozat, “Incorporating Nesterov Momentum into Adam,” in *ICLR Workshop*, 2016.
- [28] J. K. S. . K. S. Reddi, S., “On the Convergence of Adam and Beyond,” in *ICLR*, 2018.
- [29] X. Wang, H. Ishii, L. Du, P. Cheng, and J. Chen, “Privacy-preserving distributed machine learning via local randomization and admm perturbation,” *IEEE Transactions on Signal Processing*, vol. 68, pp. 4226–4241, 2020.
- [30] C. Li, P. Zhou, L. Xiong, Q. Wang, and T. Wang, “Differentially private distributed online learning,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 30, no. 8, pp. 1440–1453, 2018.
- [31] M. Wang, C. Xu, X. Chen, H. Hao, L. Zhong, and S. Yu, “Differential privacy oriented distributed online learning for mobile social video prefetching,” *IEEE Transactions on Multimedia*, vol. 21, no. 3, pp. 636–651, 2019.
- [32] I. Ahmad, S. Shahabuddin, H. Malik, E. Harjula, T. Leppänen, L. Lovén, A. Anttonen, A. H. Sodhro, M. Mahtab Alam, M. Juntti, A. Ylä-Jääski, T. Sauter, A. Gurto, M. Ylianttila, and J. Riekk, “Machine Learning Meets Communication Networks: Current Trends and Future Challenges,” *IEEE Access*, vol. 8, pp. 223418–223460, 2020.
- [33] M. S. Elbamby, C. Perfecto, C. Liu, J. Park, S. Samarakoon, X. Chen, and M. Bennis, “Wireless edge computing with latency and reliability guarantees,” *Proceedings of the IEEE*, vol. 107, no. 8, pp. 1717–1737, 2019.
- [34] J. Wang and K. D. Pham, “An Approximate Distributed Gradient Estimation Method for Networked System Optimization Under Limited Communications,” *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 50, no. 12, pp. 5142–5151, 2020.
- [35] Y. Guan and X. Ge, “Distributed attack detection and secure estimation of networked cyber-physical systems against false data injection attacks and jamming attacks,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 1, pp. 48–59, 2018.
- [36] K. Srivastava and A. Nedic, “Distributed asynchronous constrained stochastic optimization,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 772–790, 2011.

- [37] S. Boyd, A. Ghosh, B. Prabhakar, and D. Shah, “Randomized Gossip Algorithms,” *IEEE Transactions on Information Theory*, vol. 52, 2006.
- [38] D. Jakovetic, J. Xavier, and J. M. F. Moura, “Cooperative Convex Optimization in Networked Systems: Augmented Lagrangian Algorithms With Directed Gossip Communication,” *IEEE Transactions on Signal Processing*, vol. 59, no. 8, pp. 3889–3902, 2011.
- [39] M. J. Wainwright and M. I. Jordan, *Graphical Models, Exponential Families, and Variational Inference*. 2008.
- [40] S. Kar and J. M. F. Moura, “Distributed Consensus Algorithms in Sensor Networks With Imperfect Communication: Link Failures and Channel Noise,” *IEEE Transactions on Signal Processing*, vol. 57, no. 1, pp. 355–369, 2009.
- [41] T. C. Aysal, M. J. Coates, and M. G. Rabbat, “Distributed Average Consensus With Dithered Quantization,” *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4905–4918, 2008.
- [42] A. G. Dimakis, S. Kar, J. M. F. Moura, M. G. Rabbat, and A. Scaglione, “Gossip Algorithms for Distributed Signal Processing,” *Proceedings of the IEEE*, vol. 98, no. 11, pp. 1847–1864, 2010.
- [43] V. Saligrama, M. Alanyali, and O. Savas, “Distributed Detection in Sensor Networks With Packet Losses and Finite Capacity Links,” *IEEE Transactions on Signal Processing*, vol. 54, no. 11, pp. 4118–4132, 2006.
- [44] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A. Sadeghi, “DIoT: A Federated Self-learning Anomaly Detection System for IoT,” in *2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS)*, pp. 756–767, 2019.
- [45] J. Zhang, Z. Jiang, Z. Chen, and X. Hu, “WMGCN: Weighted Meta-Graph Based Graph Convolutional Networks for Representation Learning in Heterogeneous Networks,” *IEEE Access*, vol. 8, pp. 40744–40754, 2020.
- [46] C. Shi, Y. Li, J. Zhang, Y. Sun, and P. S. Yu, “A survey of heterogeneous information network analysis,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 1, pp. 17–37, 2017.
- [47] W. Choi, K. Duraisamy, R. G. Kim, J. R. Doppa, P. P. Pande, D. Marculescu, and R. Marculescu, “On-Chip Communication Network for Efficient Training of Deep Convolutional Networks on Heterogeneous Manycore Systems,” *IEEE Transactions on Computers*, vol. 67, no. 5, pp. 672–686, 2018.



- [48] M. Ma, A. Zhu, S. Guo, and Y. Yang, “Intelligent network selection algorithm for multi-service users in 5g heterogeneous network system: Nash q-learning method,” *IEEE Internet of Things Journal*, pp. 1–1, 2021.
- [49] S. Sardellitti, G. Scutari, and S. Barbarossa, “Joint optimization of radio and computational resources for multicell mobile-edge computing,” *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, no. 2, pp. 89–103, 2015.
- [50] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, “Wireless network intelligence at the edge,” *Proceedings of the IEEE*, vol. 107, no. 11, pp. 2204–2239, 2019.
- [51] A. E. Eshratifar, M. S. Abrishami, and M. Pedram, “Jointdnn: An efficient training and inference engine for intelligent mobile cloud computing services,” *IEEE Transactions on Mobile Computing*, vol. 20, no. 2, pp. 565–576, 2021.
- [52] W. Kim, M. S. Stanković, K. H. Johansson, and H. J. Kim, “A distributed support vector machine learning over wireless sensor networks,” *IEEE Transactions on Cybernetics*, vol. 45, no. 11, pp. 2599–2611, 2015.
- [53] P. Ray and P. K. Varshney, “Estimation of spatially distributed processes in wireless sensor networks with random packet loss,” *IEEE Transactions on Wireless Communications*, vol. 8, no. 6, pp. 3162–3171, 2009.
- [54] S. Jayaprakasam, S. K. A. Rahim, and C. Y. Leow, “Distributed and collaborative beamforming in wireless sensor networks: Classifications, trends, and research directions,” *IEEE Communications Surveys Tutorials*, vol. 19, no. 4, pp. 2092–2116, 2017.
- [55] Q. Zhou, D. Li, S. Kar, L. M. Huie, H. V. Poor, and S. Cui, “Learning-based distributed detection-estimation in sensor networks with unknown sensor defects,” *IEEE Transactions on Signal Processing*, vol. 65, no. 1, pp. 130–145, 2017.
- [56] P. A. Forero, A. Cano, and G. B. Giannakis, “Distributed clustering using wireless sensor networks,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 5, no. 4, pp. 707–724, 2011.
- [57] B. Polyak and Y. Tsympkin, “Pseudogradient Adaptation and Training algorithms,” *Automation and Remote Control*, vol. 34, pp. 377–397, 01 1973.
- [58] M. Eisen, A. Mokhtari, and A. Ribeiro, “Decentralized Quasi-Newton Methods,” *IEEE Transactions on Signal Processing*, vol. 65, no. 10, pp. 2613–2628, 2017.

- [59] L. Bottou, F. E. Curtis, and J. Nocedal, “Optimization Methods for Large-Scale Machine Learning,” *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.
- [60] X. Li, “Preconditioned Stochastic Gradient Descent,” *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 5, pp. 1454–1466, 2018.
- [61] M. K. Pakhira, “A linear time-complexity k-means algorithm using cluster shifting,” in *2014 International Conference on Computational Intelligence and Communication Networks*, pp. 1047–1051, Nov 2014.
- [62] N. Zong, F. Gui, and M. Adjouadi, “A new clustering algorithm of large datasets with  $o(n)$  computational complexity,” in *5th International Conference on Intelligent Systems Design and Applications (ISDA’05)*, pp. 79–82, Sep. 2005.
- [63] X. Sun, Y. Guo, Z. Liu, and S. Kimura, “A Radix-4 Partial Product Generation-Based Approximate Multiplier for High-speed and Low-power Digital Signal Processing,” in *2018 25th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, pp. 777–780, 2018.
- [64] W. Liu, L. Qian, C. Wang, H. Jiang, J. Han, and F. Lombardi, “Design of Approximate Radix-4 Booth Multipliers for Error-Tolerant Computing,” *IEEE Transactions on Computers*, vol. 66, no. 8, pp. 1435–1441, 2017.
- [65] G. Zervakis, K. Tsoumanis, S. Xydis, D. Soudris, and K. Pekmestzi, “Design-Efficient Approximate Multiplication Circuits Through Partial Product Perforation,” *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 10, pp. 3105–3117, 2016.
- [66] A. Hugeot, J. Bernard, G. Goavec-Mérou, P. Y. Bourgeois, and J. M. Friedt, “Filter Optimization for Real-Time Digital Processing of Radio Frequency Signals: Application to Oscillator Metrology,” *IEEE Transactions on Ultrasonics, Ferroelectrics, and Frequency Control*, vol. 67, no. 2, pp. 440–449, 2020.
- [67] Z. Han, M. Hong, and D. Wang, *Signal Processing and Networking for Big Data Applications*. USA: Cambridge University Press, 1st ed., 2017.
- [68] H. V. Jagadish, J. Gehrke, A. Labrinidis, Y. Papakonstantinou, J. M. Patel, R. Ramakrishnan, and C. Shahabi, “Big data and its technical challenges.,” *Commun. ACM*, vol. 57, no. 7, pp. 86–94, 2014.
- [69] G. B. Giannakis, “Signal Processing for Big Data,” in *2014 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA)*, pp. 9–9, 2014.

- [70] Y. He, F. R. Yu, N. Zhao, H. Yin, H. Yao, and R. C. Qiu, “Big data analytics in mobile cellular networks,” *IEEE Access*, vol. 4, pp. 1985–1996, 2016.
- [71] A. Sandryhaila and J. M. F. Moura, “Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure,” *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.
- [72] A. I. Tikhonyuk, S. D. Erokhin, and T. A. Chadov, “Big data application on signal processing systems,” in *2018, Systems of Signal Synchronization, Generating and Processing in Telecommunications (SYNCHROINFO)*, pp. 1–3, 2018.
- [73] Y. Du, F. Hu, L. Wang, and F. Wang, “Framework and challenges for wireless body area networks based on big data,” in *2015 IEEE International Conference on Digital Signal Processing (DSP)*, pp. 497–501, 2015.
- [74] S. Rani, S. H. Ahmed, R. Talwar, and J. Malhotra, “Can sensors collect big data? an energy-efficient big data gathering algorithm for a wsn,” *IEEE Transactions on Industrial Informatics*, vol. 13, no. 4, pp. 1961–1968, 2017.
- [75] M. Nassralla, A. M. El-Hajj, F. Baly, and Z. Dawy, “Dynamic EEG Compression Approach with Optimized Distortion Level for Mobile Health Solutions,” in *IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–5, 2016.
- [76] A. Chandra and S. Chattopadhyay, “Design of Hardware Efficient FIR Filter: A Review of the State-of-the-Art Approaches,” *Engineering Science and Technology, an International Journal*, vol. 19, no. 1, pp. 212 – 226, 2016.
- [77] J. G. Proakis and D. K. Manolakis, *Digital Signal Processing (4th Edition)*. Prentice Hall, 4 ed., 2006.
- [78] P. A. Stubberud, “A computationally Efficient Technique for Designing Frequency Sampling Filters,” *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, vol. 44, pp. 45–50, Jan 1997.
- [79] R. Y. Belorutsky and I. S. Savinykh, “Modified Technique of FIR Filter Design by the Frequency Sampling Method,” in *11th International Forum on Strategic Technology (IFOST)*, pp. 259–262, 2016.
- [80] A. V. Oppenheim, R. W. Schaffer, and J. R. Buck, *Discrete-Time Signal Processing*. Prentice-hall Englewood Cliffs, second ed., 1999.

- [81] T. Parks and J. McClellan, “Chebyshev Approximation for Nonrecursive Digital Filters with Linear Phase,” *IEEE Transactions on Circuit Theory*, vol. 19, pp. 189–194, March 1972.
- [82] L. Rabiner and O. Herrmann, “On the Design of Optimum FIR Low-Pass Filters with even Impulse Response Duration,” *IEEE Transactions on Audio and Electroacoustics*, vol. 21, pp. 329–336, August 1973.
- [83] E. Y. Remez, “General Communication Methods of Chebyshev Approximation,” *Atomic Energy Translation 4491, Kiev, U.S.S.R.*, pp. 1–85.
- [84] P. Zahradnik, “Robust Analytical Design of Optimal Equiripple Lowpass FIR Filters,” *IEEE Signal Processing Letters*, vol. 27, pp. 755–759, 2020.
- [85] X. Yang, H. Wang, K. Liu, and Y. Xiao, “Minimax and WLS Designs of Digital FIR Filters Using SOCP for Aliasing Errors Reduction in BI-DAC,” *IEEE Access*, vol. 7, pp. 11722–11735, 2019.
- [86] H. K. Kwan and J. Liang, “Minimax Design of Linear Phase FIR Filters using Cuckoo Search Algorithm,” in *8th International Conference on Wireless Communications Signal Processing (WCSP)*, pp. 1–4, Oct 2016.
- [87] J. Liang and H. K. Kwan, “FIR Filter Design using Multiobjective Cuckoo Search Algorithm,” in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, 2017.
- [88] T. Chen, Q. Wang, and H. Liu, “FIR Digital Filter Design Based on Evolutionary Multi-Objective Algorithm,” in *14th International Conference on Computational Intelligence and Security (CIS)*, pp. 349–352, Nov 2018.
- [89] H. K. Kwan, “Asymmetric FIR Filter Design using Evolutionary Optimization,” in *2017 IEEE 30th Canadian Conference on Electrical and Computer Engineering (CCECE)*, pp. 1–4, 2017.
- [90] J. H. Webb and D. C. Munson, “Design of Sparse FIR Filters using Linear Programming,” in *IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 339–342 vol.1, May 1993.
- [91] A. Jiang and H. K. Kwan, “WLS Design of Sparse FIR Digital Filters,” *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 60, no. 1, pp. 125–135, 2013.
- [92] S. Lloyd, “Least Squares Quantization in PCM,” *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

- [93] G. Ahalya and H. M. Pandey, “Data Clustering Approaches Survey and Analysis,” in *International Conference on Futuristic Trends on Computational Analysis and Knowledge Management (ABLAZE)*, pp. 532–537, Feb 2015.
- [94] M. H. Nassralla, M. M. Mansour, and L. M. A. Jalloul, “A low-complexity detection algorithm for the primary synchronization signal in lte,” *IEEE Transactions on Vehicular Technology*, vol. 65, no. 10, pp. 8751–8757, 2016.
- [95] G. Epilepsy Center of the University Hospital of Freiburg, *EEG Database*. Available at <http://epilepsy.uni-freiburg.de/freiburg-seizure-prediction-project/eeg-database>.
- [96] L. Mertz, “Sending out an SOS ? and More: Next-generation textiles and EEG headsets transport vital biomed information.,” *IEEE Pulse*, vol. 6, pp. 30–36, March 2015.
- [97] M. Nassralla, A. M. El-Hajj, F. Baly, and Z. Dawy, “Dynamic EEG compression approach with optimized distortion level for mobile health solutions,” in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–5, Sep. 2016.
- [98] G. Antoniol and P. Tonella, “EEG data compression techniques,” *IEEE Transactions on Biomedical Engineering*, vol. 44, pp. 105–114, Feb 1997.
- [99] A. Said and W. A. Pearlman, “A new, fast, and efficient image codec based on set partitioning in hierarchical trees,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 6, pp. 243–250, June 1996.
- [100] H. Daou and F. Labeau, “Dynamic Dictionary for Combined EEG Compression and Seizure Detection,” *IEEE Journal of Biomedical and Health Informatics*, vol. 18, pp. 247–256, Jan 2014.
- [101] G. Xu, J. Han, Y. Zou, and X. Zeng, “A 1.5-D Multi-Channel EEG Compression Algorithm Based on NLSPIHT,” *IEEE Signal Processing Letters*, vol. 22, pp. 1118–1122, Aug 2015.
- [102] I. Dhif, M. S. Ibraheem, L. Lambert, K. Hachicha, A. Pinna, S. Hochberg, I. Mhedhbi, and P. Garda, “A novel approach using WAAVES coder for the EEG signal compression,” in *2016 IEEE-EMBS International Conference on Biomedical and Health Informatics (BHI)*, pp. 453–456, Feb 2016.
- [103] D. Craven, B. McGinley, L. Kilmartin, M. Glavin, and E. Jones, “Compressed Sensing for Bioelectric Signals: A Review,” *IEEE Journal of Biomedical and Health Informatics*, vol. 19, pp. 529–540, March 2015.

- [104] Z. Zhang, T. Jung, S. Makeig, and B. D. Rao, “Compressed Sensing of EEG for Wireless Telemonitoring With Low Energy Consumption and Inexpensive Hardware,” *IEEE Transactions on Biomedical Engineering*, vol. 60, pp. 221–224, Jan 2013.
- [105] M. Nasrallah, A. M. El-Hajj, and Z. Dawy, “On EEG lossy data compression for data-intensive neurological mobile health solutions,” in *2015 International Conference on Advances in Biomedical Engineering (ICABME)*, pp. 309–312, Sep. 2015.
- [106] G. Higgins, S. Faul, R. P. McEvoy, B. McGinley, M. Glavin, W. P. Marnane, and E. Jones, “EEG compression using JPEG2000: How much loss is too much?,” in *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology*, pp. 614–617, Aug 2010.
- [107] M. H. Nassralla, N. Akl, and Z. Dawy, “A clustering-based approach for designing low complexity fir filters,” *IEEE Signal Processing Letters*, vol. 28, pp. 299–303, 2021.
- [108] M. Nassralla, A. M. El-Hajj, F. Baly, and Z. Dawy, “Dynamic eeg compression approach with optimized distortion level for mobile health solutions,” in *2016 IEEE 18th International Conference on e-Health Networking, Applications and Services (Healthcom)*, pp. 1–5, 2016.