

AMERICAN UNIVERSITY OF BEIRUT

DISTRIBUTED LOGISTIC CLASSIFIERS IN  
SEMI-SUPERVISED SETTINGS

by  
MAHA GERGES ISSA

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

Beirut, Lebanon  
August 2022

# AMERICAN UNIVERSITY OF BEIRUT

## DISTRIBUTED LOGISTIC CLASSIFIERS IN SEMI-SUPERVISED SETTINGS

by  
MAHA GERGES ISSA

Approved by:

---

Dr. Roula Nassif, Assistant Professor  
Electrical and Computer Engineering

Advisor



---

Dr. Ibrahim Issa, Assistant Professor  
Electrical and Computer Engineering

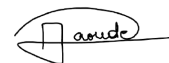
Member of Committee

*Ibrahim Issa*

---

Dr. Dany Abou Jaoude, Assistant Professor  
Mechanical Engineering

Member of Committee



Dany Abou  
Jaoude  
2022.09.12  
20:49:38 +03'00'

Date of thesis defense: August 11, 2022

# AMERICAN UNIVERSITY OF BEIRUT

## THESIS RELEASE FORM

Student Name: Issa Maha Gerges  
Last First Middle

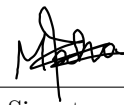
I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of my thesis; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes

--- As of the date of submission of my thesis

After 1 year from the date of submission of my thesis .

--- After 2 years from the date of submission of my thesis .

--- After 3 years from the date of submission of my thesis .



Signature

15/9/2022

Date

# ACKNOWLEDGEMENTS

I would like to disclose my sincere acknowledgment to everyone who have helped and supported me in the accomplishment of this thesis.

First, my deepest gratitude goes to Dr. Roula Nassif who did me the honor of accepting to be my thesis advisor. She always made herself available throughout the research process of this thesis and during our preparation of a conference publication based on the thesis research work, in collaboration with Ms. Elsa Rizk and Prof. Ali H. Sayed. Dr. Nassif did not also hesitate in guiding me during the drafting of this report. I am extremely fortunate to have received her continuous guidance and support.

I am also grateful to my thesis committee members, Dr. Ibrahim Issa and Dr. Dany Abou Jaoude, for providing their constructive feedback that helped me revise and edit my work.

Additionally, I would like to express my sincere appreciation to the American University of Beirut (AUB) Faculty and Staff for their efforts during my two-year journey.

Finally, I would like to thank my amazing AUB colleagues who never hesitated to stand by my side throughout all the stages of our academic journey. I also wish to express my gratefulness to my family and friends for their contribution, support, and encouragement.

# ABSTRACT OF THE THESIS OF

Maha Gerges Issa for Master of Engineering  
Major: Electrical and Computer Engineering

Title: Distributed Logistic Classifiers in Semi-Supervised Settings

In network semi-supervised learning problems, only a subset of the network nodes is able to access the data labeling. This thesis formulates a decentralized optimization problem where agents represent classifiers that may observe different numbers and types of features, and hence have individual decision rules to estimate, subject to the condition that neighboring agents are more likely to have similar labels. To promote such relationships, we propose to add to the individual logistic regression costs a graph regularization term that allows to penalize the differences between the labels at neighboring agents. Two regularization terms are investigated: a sparsity promoting regularizer and a smoothness promoting regularizer. Streaming data is assumed, and therefore, the *stochastic* (sub-)gradient descent method is used to solve the regularized problem. We provide some important assumptions and conditions that guarantee the stability and convergence of the proposed algorithm in the mean-square-error sense. Simulation results show that collaboration among neighboring agents, which is promoted through the added regularization term, can lead to better classification results by decreasing the probability of error and by improving the convergence rate. Those results are promising in semi-supervised settings where some agents do not have access to labeled data points due to cost or privacy reasons, and in applications with limited amount of data.

# TABLE OF CONTENTS

<b>ACKNOWLEDGEMENTS</b>	<b>1</b>
<b>ABSTRACT</b>	<b>2</b>
<b>ABBREVIATIONS</b>	<b>7</b>
<b>NOTATIONS</b>	<b>8</b>
<b>1 Introduction</b>	<b>9</b>
1.1 Signal Processing on Graphs . . . . .	9
1.2 Decentralized Inference Over Graphs . . . . .	9
1.3 Single-Task Estimation Problems . . . . .	10
1.4 Multitask Estimation Problems . . . . .	10
1.5 Heterogeneous Settings . . . . .	11
1.5.1 Heterogeneous setting 1: different types and numbers of features	11
1.5.2 Heterogeneous setting 2: semi-supervised setting . . . . .	11
1.6 Problem Statement . . . . .	12
1.7 Thesis Outline . . . . .	13
<b>2 Literature Review</b>	<b>14</b>
2.1 Cooperative Strategies for Solving Network Estimation Problems . . . .	14
2.1.1 Cooperation in single-task estimation problems . . . . .	14
2.1.2 Cooperation in multitask estimation problems to promote pa- parameter vectors smoothness . . . . .	15
2.1.3 Cooperation in multitask estimation problems to promote graph clustering . . . . .	15
2.1.4 Cooperation in multitask estimation problems to promote graph piecewise constant transitions . . . . .	15
2.1.5 Cooperation in multitask estimation problems to promote agents' predictions smoothness . . . . .	16
2.2 Cooperative Strategies for Solving Network Classification Problems in Semi-Supervised Settings . . . . .	16
2.2.1 Cooperation in semi-supervised learning problems where nodes represent single data points . . . . .	16

2.2.2	Cooperation in semi-supervised learning problems where nodes carry local datasets . . . . .	17
2.3	Comparison to Our Approach . . . . .	18
<b>3</b>	<b>Methodology</b>	<b>21</b>
3.1	Initial Problem Definition . . . . .	21
3.2	Additional Regularizers . . . . .	22
3.2.1	Network Lasso regularization . . . . .	22
3.2.2	Graph Laplacian regularization . . . . .	23
3.3	Regularized Optimization Problem . . . . .	23
3.4	Decentralized Semi-Supervised Multitask Learning Algorithm [1] . . . .	24
<b>4</b>	<b>Stability Analysis</b>	<b>27</b>
4.1	Network Error Vector Recursion . . . . .	27
4.2	Assumptions . . . . .	29
4.3	Network Mean-Square-Error Recursion . . . . .	29
4.4	Network Mean Error Recursion . . . . .	32
4.5	Network Mean-Square-Error Stability . . . . .	33
<b>5</b>	<b>Experimental Results</b>	<b>35</b>
5.1	Synthetic Data Experiments . . . . .	35
5.1.1	Experiment 1 . . . . .	36
5.1.2	Experiment 2 . . . . .	37
5.2	Real Data Experiments . . . . .	38
5.2.1	Experiment 1 . . . . .	41
5.2.2	Experiment 2 . . . . .	43
<b>6</b>	<b>Conclusion</b>	<b>45</b>
6.1	Conclusion . . . . .	45
6.2	Future Research Work . . . . .	45
	<b>Bibliography</b>	<b>47</b>

# ILLUSTRATIONS

1.1	Random graph with 15 nodes where each node is connected by an edge to four others, resulting in 30 edges. . . . .	10
1.2	Illustration of the two sources of heterogeneities in the data acquisition process. . . . .	12
3.1	Sign function $\text{sign}(x)$ and its hyperbolic tangent approximations $\tanh(cx)$ for several values of $c$ [1]. . . . .	24
5.1	Clustered network structure (agents with the same color observe the same label) [1]. . . . .	36
5.2	Synthetic data experiment 1: Network average validation loss while fixing the informed nodes and using the network Lasso regularization promoting sparsity ( $f(x) =  x $ ) [1]. . . . .	37
5.3	Synthetic data experiment 1: Network average validation loss while fixing the informed nodes and using the graph Laplacian regularization promoting smoothness ( $f(x) = x^2$ ) [1]. . . . .	38
5.4	Synthetic data experiment 2: Network average validation loss while adopting the random sampling and using the network Lasso regularization promoting sparsity ( $f(x) =  x $ ) [1]. . . . .	39
5.5	Synthetic data experiment 2: Network average validation loss while adopting the random sampling and using the graph Laplacian regularization promoting smoothness ( $f(x) = x^2$ ) [1]. . . . .	39
5.6	Illustration of the 139 weather stations (or network nodes) with the informed nodes painted in black (with probability $q_k = 1$ ) and spread across the network. . . . .	42
5.7	Real data experiment 1: Illustration of the 139 weather stations along with their class labels on March 13, 2014. The black color corresponds to the label 1 (rainy or snowy day) whereas the copper color represents the label $-1$ (non-rainy and non-snowy day). . . . .	43



# TABLES

5.1	Real data experiment 1: Network average testing error while fixing the informed nodes and using the network Lasso regularization promoting sparsity ( $f(x) =  x $ ). . . . .	41
5.2	Real data experiment 1: Network average testing error while fixing the informed nodes and using the graph Laplacian regularization promoting smoothness ( $f(x) = x^2$ ). . . . .	41
5.3	Real data experiment 2: Network average testing error while adopting the random sampling and using the network Lasso regularization promoting sparsity ( $f(x) =  x $ ). . . . .	44
5.4	Real data experiment 2: Network average testing error while adopting the random sampling and using the graph Laplacian regularization promoting smoothness ( $f(x) = x^2$ ). . . . .	44

# ABBREVIATIONS

Lasso	Least Absolute Shrinkage and Selection Operator
MSE	Mean Squared Error
SBM	Stochastic Block Model

# NOTATIONS

Lower-case letters	Column vectors and scalars
Upper-case letters	Matrices
Boldface letters	Random quantities
Normal font letters	Deterministic quantities
$(\cdot)^\top$	Matrix or vector transposition
$\ \cdot\ $	2-norm of a matrix or Euclidean norm ( $\ell_2$ -norm) of a vector
$\ \cdot\ _1$	$\ell_1$ -norm of a vector
$ \cdot $	$\ell_1$ -norm (absolute value) of a scalar
$\text{col}\{\cdot\}$	Operator stacking the column vector entries on top of each other
$\text{diag}\{\cdot\}$	Operator creating a matrix by inserting each block argument below and to the right of the one that precedes it

# CHAPTER 1

## INTRODUCTION

This introductory chapter, which contains parts from the introduction in [1], aims at giving an overview of the field of this thesis. It also presents the study context of this work by providing the motivations and the problem statement. Finally, an outline of the organization of this thesis work is described.

### 1.1 Signal Processing on Graphs

Recently, the area of signal processing on graphs has been extensively explored due to its various applications [2]. Graphs, which are common representations of data, can be seen as a network containing several nodes where each node can represent a data point or a process collecting data in a continuous manner. An example of such networks or graphs is represented in Figure 1.1. As illustrated in this figure, graph nodes can be connected by edges that are associated with weights reflecting the similarities between these nodes. A network, such as the one depicted in Figure 1.1, is represented by a graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, A\}$ , where  $\mathcal{V} = \{1, \dots, N\}$  denotes the set of nodes (where  $N$  is the total number of nodes in the network),  $\mathcal{E}$  denotes the set of edges, and  $A$  is the adjacency matrix that contains the weights of the edges. The neighborhood of node or agent  $k$  consists of all the agents that are connected to  $k$  by an edge, and is denoted by  $\mathcal{N}_k$ . Network structured problems appear in several domains such as transportation networks, brain imaging, image processing, and statistical and machine learning problems [2].

### 1.2 Decentralized Inference Over Graphs

Prior works have exploited the graph connectivity to solve different network estimation and classification problems in a decentralized manner, assuming that agents can communicate over the graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, A\}$  [3]–[6]. This is referred to as decentralized inference over graphs, which has received considerable attention over the past two decades. In a decentralized implementation, and instead of establishing a central processor that collects data from all the nodes to perform the optimization task, each node estimates its decision rule locally. In other words, in a decentralized

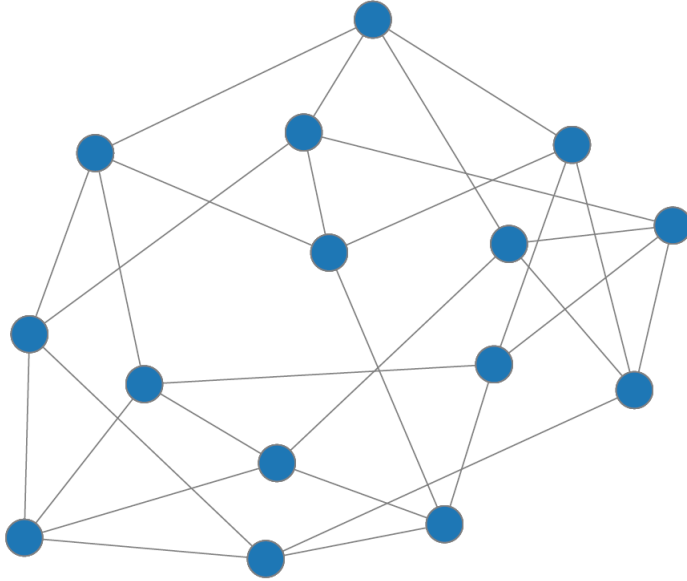


Figure 1.1: Random graph with 15 nodes where each node is connected by an edge to four others, resulting in 30 edges.

learning setting, each agent minimizes its own risk function by performing local computations and collaborates with its neighboring agents by exchanging some estimates with them. Compared to the non-cooperative approaches, appropriately designed decentralized cooperative strategies have been shown to achieve better results in terms of network performance [3], [4], [7].

### 1.3 Single-Task Estimation Problems

With some exceptions, most of the works on decentralized inference over graphs focus on *consensus* and *diffusion* optimization by considering variations of this problem [7]–[9]:

$$w^o = \underset{w}{\operatorname{argmin}} \sum_{k=1}^N J_k(w) \quad (1.1)$$

where  $J_k(w)$  represents a private cost function at agent  $k$  that depends on an  $M$ -dimensional parameter vector  $w \in \mathbb{R}^M$ . Such problem is referred to as a single-task estimation problem since all the  $N$  agents seek to estimate the same parameter vector  $w^o$  in (1.1) and reach consensus.

### 1.4 Multitask Estimation Problems

In modern machine learning applications, agents generate data in a highly non-identically distributed manner. Such networks require more complex models and flexible algorithms than traditional single-task implementations since their agents may need to simultaneously estimate and track distinct tasks or objectives. Such

problems are referred to as multitask estimation problems. However, the tasks or parameter vectors that agents seek to estimate can sometimes be related. Previous efforts in this direction attempted to solve *variations* of the following regularized multitask learning formulation [4]:

$$w^* = \underset{w}{\operatorname{argmin}} \sum_{k=1}^N J_k(w_k) + \eta \mathcal{R}(w_1, \dots, w_N) \quad (1.2)$$

where  $w = \operatorname{col}\{w_1, \dots, w_N\}$  denotes the collection of parameter vectors across the network,  $w_k \in \mathbb{R}^M$  is the parameter vector or task at agent  $k$ ,  $\eta > 0$  is a regularization strength, and  $\mathcal{R}(\cdot)$  is a regularization function promoting the relationships between the tasks.

## 1.5 Heterogeneous Settings

The data acquisition process in machine learning applications can be affected by several sources of heterogeneity. Such heterogeneities can have undesirable effects on the learning process of the machine learning models. In the sequel, we present two examples of heterogeneous settings.

### 1.5.1 *Heterogeneous setting 1: different types and numbers of features*

Assuming that network agents represent machine learning models, every agent should have access to a set of features or attributes at every time instant. These features act as input variables to every machine learning model and are essential for optimizing the model. In a heterogeneous system setting, as in [10], agents might have access to different types and numbers of features. For instance, in weather sensor networks, some sensors might observe measurements related to wind speed, temperature, etc., whereas others might not have access to wind speed measurements or might observe another set of features. An example of such heterogeneous setting is depicted in Figure 1.2, where entries of the feature vectors with similar color correspond to the same observed feature type. It is noticed that the length of the feature vectors and the colors of their components vary between agents. Such heterogeneities can lead to “different” learning abilities across the network. In other words, agents that observe a small number of (relevant) features have limited learning abilities in comparison with agents that have access to a large number of (relevant) features.

### 1.5.2 *Heterogeneous setting 2: semi-supervised setting*

Besides features types and numbers, another source of heterogeneity in the data acquisition process can appear in machine learning problems, and particularly in classification problems. Some agents in the network might have access to unlabeled data points at a given time instant. This scenario arises in applications where labeling at some agents is costly and requires human assistance, or in other applications where some agents are not willing to use their own labels in the training process

due to privacy concerns. This is referred to as semi-supervised learning, which is used in machine learning applications where only a part of the data is labeled. An illustration of this second heterogeneous setting is also shown in Figure 1.2, where agents with the same color are observing the same label, and agents with a gray color are unlabeled. This prevents these agents that have access to unlabeled data points from properly learning their classification rules.

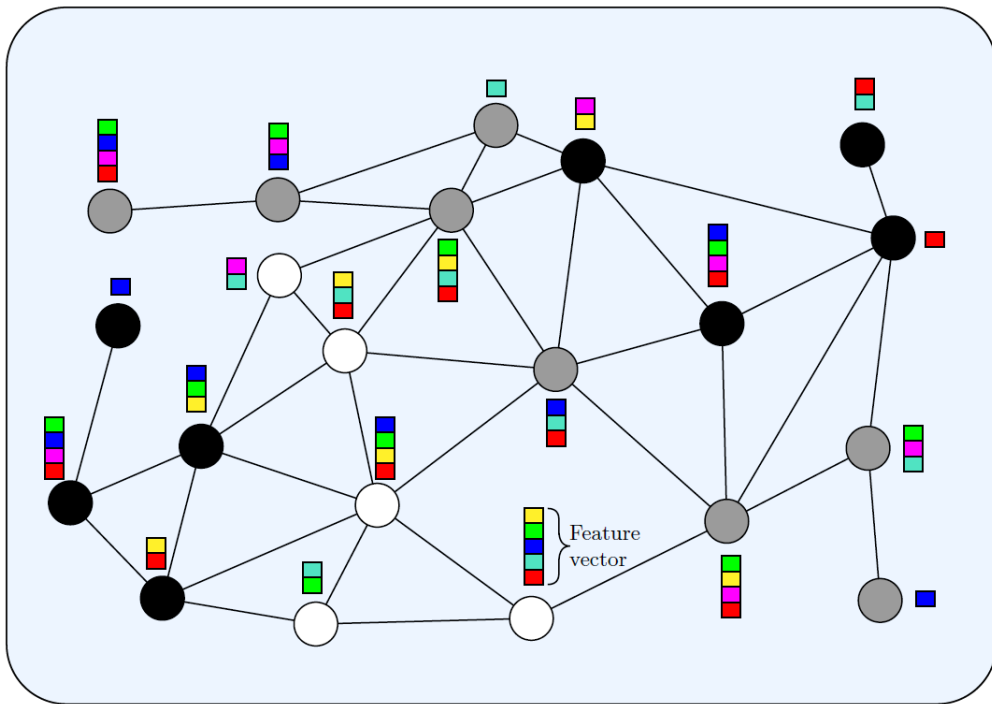


Figure 1.2: Illustration of the two sources of heterogeneities in the data acquisition process.

## 1.6 Problem Statement

In this study, we focus on decentralized semi-supervised multitask learning in streaming and heterogeneous data acquisition settings, where we consider the two sources of heterogeneities that are illustrated in Figure 1.2: different types and numbers of features and the semi-supervised setting. Each agent  $k$  is collecting at each time instant  $i$  an  $M_k \times 1$  feature vector  $\mathbf{h}_{k,i} \in \mathbb{R}^{M_k}$ , which corresponds to a collection of observed attributes in a binary classification problem (where there exists only two classes), and is interested in estimating its own decision rule  $w_k^o$ . Agents with a small number of observed attributes or agents that have access to unlabeled data points can considerably benefit from cooperating with their neighbors since, in some applications, they are more likely to observe similar labels. For instance, in weather forecasting applications [11], if it is raining in a given city on a given day, it is more likely that rain is also occurring in adjacent cities on this same day. However, problem formulation (1.2) might not lead to a meaningful cooperation rule since,

in our considered setting, we need to promote the relationships between the labels  $\{\gamma_k\}$ , instead of promoting the relationships between the tasks or parameter vectors  $\{w_k\}$ . To promote the labels' relationships, we propose to add to the cost function  $J_k(w_k)$  a regularization term consisting of a weighted sum of  $\ell_1$ -norms (or squared  $\ell_2$ -norms) of the differences between the labels.

## 1.7 Thesis Outline

The remainder of this thesis report is organized as follows:

- Chapter 2: This chapter summarizes previous works in the literature that are related to this thesis. For each work, we briefly mention the tackled problem, its adopted method, and its most important findings. We start by listing in section 2.1 several cooperative strategies that were proposed to solve network estimation problems, and then move to network semi-supervised learning problems in section 2.2. Finally, we give a detailed comparison between previous works and our work to provide a clearer idea about its novelty (section 2.3).
- Chapter 3: In this chapter, we explain the methods that we employed to solve our problem. First, we clearly define the objective of our problem in section 3.1. Then, we provide the mathematical definitions of our two proposed additional regularization terms in section 3.2. Section 3.3 gives the new regularized problem, and finally section 3.4 explains how we solve it.
- Chapter 4: This chapter studies the behavior of our proposed algorithm. We start by deriving the network error vector recursion in section 4.1. Then, we list some useful assumptions in section 4.2. Sections 4.3 and 4.4 derive the network mean-square-error recursion and the network mean error recursion, respectively. Finally, the algorithm stability in the mean-square-error sense is examined in section 4.5.
- Chapter 5: To demonstrate the advantages of our approach, we present in this chapter results pertaining to synthetic data experiments (section 5.1) in addition to real data experiments (section 5.2), while discussing them and showing their significance.
- Chapter 6: Finally, this chapter concludes this thesis work by providing a brief summary (section 6.1) and some recommendations that may improve future research (section 6.2).



# CHAPTER 2

## LITERATURE REVIEW

This chapter provides a literature survey that gives a general overview about some important concepts related to this thesis work. We start by listing several works that proposed cooperative strategies between agents to help solve network estimation problems. Then, we summarize various works that targeted network semi-supervised learning problems. Finally, we provide a comparison that demonstrates how our proposed approach is different from previous works in terms of the considered settings and the proposed methods.

### 2.1 Cooperative Strategies for Solving Network Estimation Problems

In a network structured estimation problem, the cooperation among neighboring agents in the graph can be formulated in different ways. In this section, we present several works that targeted such problems.

#### 2.1.1 *Cooperation in single-task estimation problems*

In single-task estimation problems, i.e., when the agents that are observing the same number of features are interested in estimating the same parameter vector or decision rule, diffusion strategies can be used [3]. An example of such strategies is the Adapt-then-Combine (ATC) diffusion strategy that was proposed for distributed online learners in an online machine learning problem. In this case, and at each iteration  $i$ , each agent  $k$  first applies a stochastic gradient descent update to minimize its own risk function. Then, the resulting intermediate estimates are exchanged between neighboring nodes and are properly combined to obtain an updated estimate of the decision rule at iteration  $i$  and at each agent  $k$ . Such strategies were shown to ensure the convergence of each agent in the network to the solution that minimizes the global objective function in (1.1), and at a similar rate to the centralized implementation.

### ***2.1.2 Cooperation in multitask estimation problems to promote parameter vectors smoothness***

In multitask estimation problems, the agents are interested in estimating different, though related, parameter vectors. In such case, properly designed multitask based approaches can be used [4], and a regularization term can be added to the global objective function to be minimized, as in (1.2). The aim of this regularization is to encourage the relationships between the parameter vectors. One example of such regularizer is the graph Laplacian regularization that is used to promote the smoothness of a signal over the graph [5], [11], [12]. For instance, a multitask network was considered in [5], where each agent is interested in estimating its own parameter vector under the prior knowledge that this vector varies smoothly over the graph. To solve this problem, a multitask version of the diffusion strategy [3] was proposed where the stochastic gradient descent algorithm was also used, but the cooperation rule was updated based on the use of the graph Laplacian regularizer to promote the smoothness of the parameter vectors, which are considered the graph signal in this case.

### ***2.1.3 Cooperation in multitask estimation problems to promote graph clustering***

Another regularization term can be added to the global multitask optimization problem and is the extension of the Least Absolute Shrinkage and Selection Operator (Lasso) to network structured problems. This regularizer is referred to as the network Lasso regularizer, which is employed to promote the graph clustering [6]. In this case, a weighted sum of  $\ell_2$ -norms of the differences between the parameter vectors is added to the global loss function. The advantage of using this regularization instead of the previously discussed graph Laplacian regularizer is encouraging parameter vectors at strongly connected nodes to be equal. This divides the network into many clusters, where agents in the same cluster have a common solution for the parameter vectors. This problem was solved by using an algorithm based on the Alternating Direction Method of Multipliers (ADMM). The proposed approach was tested on several problems and has demonstrated to achieve better results in comparison with the unregularized frameworks: A higher accuracy was reached in a classification example and a lower Mean Squared Error (MSE) was attained in a regression example (where the target is to predict a real value instead of a label).

### ***2.1.4 Cooperation in multitask estimation problems to promote graph piecewise constant transitions***

In some applications, the optimal solutions of the parameter vectors for adjacent nodes may share a large number of common entries. To handle such situations, a new regularization term was suggested in [13] and [14] that is based on a weighted sum of  $\ell_1$ -norms of the differences between the parameter vectors. Such regularizer encourages sparsity and hence can promote the piecewise constant transitions of the parameter vectors over the graph. Due to the non-differentiability of this ad-

ditional regularizer, the proposed approach to solve this problem was based on the subgradient method in [14], while in [13] it was based on the proximal projection operator where a closed form expression was derived to evaluate this operator for a better efficiency. Simulation results in both [13] and [14] showed that cooperation between agents, which is incorporated in the proposed regularization, can decrease the network Mean Squared Deviation (MSD) and hence can lead to an improved network performance in such scenarios.

### **2.1.5 *Cooperation in multitask estimation problems to promote agents' predictions smoothness***

In heterogeneous machine learning applications where agents may observe different numbers of features, combining or forcing smoothness on the parameter vectors to be estimated is not possible. Furthermore, differences between these parameter vectors cannot be computed. This is because the feature and parameter vectors have the same dimension, which implies that the dimension of the parameter vectors is inconsistent across the graph.

To solve network binary classification problems in such heterogeneous setting, the graph Laplacian regularization was proposed in [10] to promote the graph signal smoothness, however, the graph signal considered in this case is the predicted output of the agents or classifiers. This output pertains to the inner product of the feature and parameter vectors. This proposed approach was introduced under the prior information that labels at neighboring classifiers are more likely to be the same. The problem was solved using the stochastic gradient descent algorithm and the simulation results proved that the cooperative scenarios involving the proposed regularization helped decrease the network average testing error.

## **2.2 Cooperative Strategies for Solving Network Classification Problems in Semi-Supervised Settings**

Cooperation among network agents has also been widely exploited to solve network semi-supervised learning problems and has achieved good results [15]–[23]. This section lists several works that have targeted such problems where agents in the graph can either represent an individual data point or a classifier carrying a local dataset.

### **2.2.1 *Cooperation in semi-supervised learning problems where nodes represent single data points***

The problem of labeling a partially labeled graph has been considered in the machine learning community, with the so-called *semi-supervised learning on graphs* [15]–[18]. Within this community, each node in the network represents a single data point, the graph connections represent the similarities between these data points or their feature vectors, and closer data points tend to have similar class labels. For instance, in survey sampling, instead of surveying the whole population, only a subset of this

population can be selected to be surveyed and the remaining peoples' preferences are inferred based on features' similarities [15].

Several strategies have been derived to classify partially labeled datasets under the assumption that all data are available beforehand [15]–[21]. In [15] and [16], the graph Laplacian regularization was used to solve the partially labeled datasets classification problem, while in [17] a label propagation algorithm was derived. In [19] and [20], the loss function to be minimized in the classification problem was formulated as the sum of the empirical losses at the labeled nodes in addition to a non-smooth total variation regularization based on the network Lasso. In [20], the total variation term corresponds to a weighted sum of the  $\ell_2$ -norms of the differences between the parameters to be estimated, whereas in [19], it corresponds to a weighted sum of the  $\ell_1$ -norms of the differences between the log odds ratios, which are calculated as:

$$x_k = \log \frac{\mathbb{P}\{\gamma_k = 1\}}{\mathbb{P}\{\gamma_k = -1\}}, \quad (2.1)$$

where the scalar quantity  $x_k$  represents the log odds ratio at agent  $k$  and  $\gamma_k$  is the label at agent  $k$  (which can be either 1 or  $-1$ ). Such non-smooth graph-based regularization in [19] is used to encourage the sparsity of the differences between the log odds ratios at neighboring nodes. In [21], a common model was studied where the network was partitioned into two clusters, and the aim was to assign each unlabeled node to one of the two clusters. For this purpose, a convex optimization problem, which could also be formulated as a linear program, was solved by minimizing the total variation of the indicator graph signal, which is a vector at each node indicating to which cluster this node belongs (if the node belongs to the  $j$ -th cluster then the  $j$ -th entry of this node's vector is 1 and the remaining entry is 0).

### 2.2.2 *Cooperation in semi-supervised learning problems where nodes carry local datasets*

In applications where devices generate local data samples, each device can be modelled as an agent in a network carrying a local dataset, instead of representing a single data point. Such settings have also been considered in the machine learning community, and are referred to as *networked federated learning* [22], [23]. In [22], each network node carries a local dataset that is also supposed to be available in advance, and is interested in finding the parameter vector of its own model. The network was assumed to be divided into several clusters and nodes within the same cluster were supposed to have similar parameter vectors. To model applications where accessing data may be costly, only a part of the local datasets was used for the training process. To recompense this, a regularization function was added to the training error of nodes within the training set. This regularization consisted of an increasing function of the parameters' differences to encourage their equality for nodes within the same cluster. This proposed networked federated multitask learning algorithm was solved using the primal-dual method. The work in [23] is similar to [22], but it specifically considered the  $\ell_1$ -norm function in the additional regularization, which has led to a network Lasso formulation for the problem. Simulation

results showed that the proposed approach helped decrease the network MSE.

## 2.3 Comparison to Our Approach

This section is dedicated to explain the differences between the approach that we are proposing in this thesis work and the previous methods in the literature that we already summarized. First, our work handles cases where each agent in the network aims to estimate its own parameter vector, which makes it different from the diffusion strategies in single-task problems [3] (subsection 2.1.1) where agents need to agree on a common solution. In the sequel, we focus on the differences between our work and the previous methods in multitask learning and in semi-supervised settings, particularly [11], [6], [10], [20], and [23]. The comparison that we are conducting is based on four criteria:

- The parameter vectors dimensionality: whether they have the same dimension for all the agents or their dimension is inconsistent over the graph (heterogeneous setting 1)
- The data acquisition setting: whether the data are available beforehand or are acquired in a streaming manner
- The data labeling availability: whether the setting is supervised or semi-supervised (heterogeneous setting 2), and in the latter case whether each agent represents an individual data point or carries a local dataset
- The proposed additional regularization term: whether it is based on the network Lasso or the graph Laplacian, in addition to the argument inside the regularization function (parameter vectors, inner products of the parameter and feature vectors, or labels)

Starting with the data acquisition setting, our proposed approach responds to streaming data, while most of the previous works, with few exceptions, assume that the data are available in advance. Approaches that respond to online streaming data have the advantage that they can continuously learn and track the solution of the problem when drifts in the data may occur. The works [11] and [10] consider streaming data settings, however, they differ from our work in several ways, which we explain in the sequel.

Moving to the parameter vectors dimensionality, four out of the five works that are considered in this comparison assume that parameter vectors have the same size for all the network agents, except [10] which considered that the dimensions of the parameter vectors might be different. Such setting, which we referred to as the heterogeneous setting 1 (subsection 1.5.1), allows for a better flexibility since it can handle more complex problems where agents might have access to different numbers and types of data measurements. In our work, we also consider this heterogeneous setting, however, as previously stated, our approach is different from [10] and the difference is clarified in the following paragraphs.

Considering now the data labeling availability, the works [11], [6], and [10] suppose that all the network agents are continuously able to access the labels of their data samples. This makes our approach different from [11] and [10] since, in our case, we consider the heterogeneous setting 2 (subsection 1.5.2) where only a part of the data samples is labeled. This semi-supervised setting is more suitable in several applications (for instance, in applications where data labeling may be expensive). The authors in [20] and [23] also handled semi-supervised settings, however, the purpose of the work in [20] was to label a partially labeled dataset, and hence they considered each network agent as an individual data point in the dataset. This makes our work different from [20] since, in our problem, each agent is a classifier observing several data points and the aim is to learn the decision rules of these classifiers. Finally, the difference between [23] and our approach lies in the parameter vectors dimensionality, the data acquisition setting, and the proposed regularization (which is lastly discussed).

The last criteria that we include in our comparison is the proposed additional regularization term, in addition to the argument inside the regularizer. Regardless of the function being used, most of the previous literature incorporated the parameter vectors inside the regularization. The choice of the function depended on the prior knowledge that the authors wished to promote, which are smoothness in the case of the graph Laplacian regularization and clustering in the case of the network Lasso regularization. However, in our work, promoting smoothness or clustering of the parameter vectors is infeasible due to the inconsistency in their dimensions. The work [10], which also considered such inconsistency, proposed to solve this issue by encouraging the smoothness of the inner products of the parameter and feature vectors, which are scalar values signaling the labels of the data samples. In other words, if this inner product is positive (or negative), then the considered data sample belongs to the positive (or negative) class. Even though the simulations in [10] showed promising results, such approach may confront problems in applications where the features are not in the same scale. For example, the inner product of an agent receiving a temperature measurement (in degree celsius) might be a small number, in contrast with the inner product of an agent only having access to the altitude (in meters), which might be a large number. In such case, these numbers do not have smooth transitions between these two agents, however, if they were of the same sign, then the two agents are observing data arising from the same class. In order to avoid such issues, usual approaches in machine learning are to rescale the data so that no feature dominates the others. Examples of such approaches include data normalization, which shifts all the features to the  $[0, 1]$  interval, and data standardization, which transforms the features in a way that they all have a zero mean and a unit variance. Such techniques are powerful and can lead to a better model performance, however, they require statistical properties of the data, such as the maximum, minimum, mean, and standard deviation. In online streaming settings, such as in [10] and our work, these statistical properties are not available beforehand, which rules out the use of normalization and standardization techniques. Hence, to overcome this scaling problem, we propose in our work to incorporate the predicted labels in the regularization function, which are either equal to 1 or  $-1$

since we are dealing with a binary classification problem. We propose both the network Lasso regularization, by using the  $\ell_1$ -norms of the differences between the labels, and the graph Laplacian regularization, by using the squared  $\ell_2$ -norms of the differences between the labels.

# CHAPTER 3

## METHODOLOGY

In this chapter, which appears in the second section of [1], we detail the methodology steps that have led to our proposed algorithm. We start by explaining the initial problem, then, we define the proposed additional regularization terms. We then move to establish the new regularized problem, and finally, we present the resulting decentralized semi-supervised multitask learning algorithm.

### 3.1 Initial Problem Definition

We consider an undirected weighted graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, A\}$ . This network consists of  $N = |\mathcal{V}|$  nodes, where each node  $k$  represents a logistic regression classifier interested in solving an online binary classification problem. Therefore, each agent  $k$  is collecting data in a continuous manner. We first assume the supervised setting where, at every time instant  $i$ , it is observing a feature vector  $\mathbf{h}_{k,i} \in \mathbb{R}^{M_k}$  and the corresponding class label  $\gamma_k(i) \in \{-1, 1\}$ . The objective is to construct a classifier at each agent  $k$  to predict the label  $\gamma_k$  based on the knowledge of the feature vector  $\mathbf{h}_k$ . To that end, each agent  $k$  can use a logistic regression machine [7], [24], [25] that seeks an  $M_k \times 1$  vector  $w_k^o$ , such that the predicted label at agent  $k$  and time instant  $i$  is evaluated as:

$$\hat{\gamma}_k(i) = \text{sign}(\mathbf{h}_{k,i}^\top w_k^o), \quad (3.1)$$

where the operator  $\text{sign}(\cdot)$  is the sign function and is evaluated for a scalar argument  $x$  as follows:

$$\text{sign}(x) = \begin{cases} +1, & \text{if } x > 0 \\ 0, & \text{if } x = 0 \\ -1, & \text{if } x < 0 \end{cases}. \quad (3.2)$$

The vector  $w_k^o$  to be estimated is the minimizer of a risk function  $J_k(w_k)$ , namely:

$$w_k^o \triangleq \underset{w_k}{\text{argmin}} J_k(w_k), \quad (3.3)$$

with

$$J_k(w_k) \triangleq \mathbb{E} \ln \left( 1 + \exp \left( - \gamma_k(i) \mathbf{h}_{k,i}^\top w_k \right) \right) + \frac{\rho}{2} \|w_k\|^2, \quad (3.4)$$



where the expectation in (3.4) is computed over the distributions of the random data  $\{\mathbf{h}_{k,i}, \gamma_k(i)\}$  and where  $\rho$  is a positive parameter that controls the importance of the relaxation term  $\|w_k\|^2 = w_k^\top w_k$ .

As previously stated, we consider that nearby classifiers are more likely to observe the same label at each time instant  $i$ . The adjacency matrix  $A$  describes the graph structure and connections. It is an  $N \times N$  symmetric matrix with the  $(k, \ell)$ -th entry  $a_{k\ell} \geq 0$  reflecting the strength of the relation between nodes  $k$  and  $\ell$ . For instance, if node  $\ell$  is connected to node  $k$  (i.e.,  $\ell \in \mathcal{N}_k$ ) and is more likely to observe the same label as node  $k$ , then the weight  $a_{k\ell}$  should be large. If there is no edge connecting nodes  $k$  and  $\ell$ , then  $a_{k\ell} = 0$ .

## 3.2 Additional Regularizers

In heterogeneous data acquisition settings, the number of observed features differs between agents. In such case, agents with a small number of (relevant) features are probably not able to make decisions on their own. Additionally, even if the number of observed features is enough, some agents may be affected by noise, which prevents them from properly estimating their decision rules. Furthermore, some agents in these heterogeneous settings may not have access to their true labels, which also hinders the estimation tasks of these agents. These facts motivate us to find a meaningful cooperation rule that allows agents with limited learning abilities to benefit from the learning process of their neighbors. To that end, two regularization functions are investigated in the following: the network Lasso regularization and the graph Laplacian regularization [1].

### 3.2.1 Network Lasso regularization

If we let

$$\widehat{\mathbf{\Gamma}}_i = \text{col}\{\widehat{\gamma}_1(i), \dots, \widehat{\gamma}_N(i)\} \quad (3.5)$$

denote the collection of predicted labels from all the network nodes, we can derive the cooperation rule by formulating a regularized optimization problem that employs the total variation of the graph signal  $\widehat{\mathbf{\Gamma}}_i$  [26]

$$\text{TV}(\widehat{\mathbf{\Gamma}}_i) = \sum_{(k,\ell) \in \mathcal{E}} a_{k\ell} |\widehat{\gamma}_k(i) - \widehat{\gamma}_\ell(i)| \quad (3.6)$$

as a regularizer. A small total variation is expected over the graph since the agents that are observing the same label are more likely to be connected by an edge with a large weight  $a_{k\ell}$ , in contrast with agents that are observing distinct labels. Hence, incorporating this total variation term into the problem formulation can improve the network performance. The non-smooth regularizer (3.6) is suitable for clustered-network applications where agents are decomposed into clusters, and within each cluster, agents are observing the same label.

### 3.2.2 Graph Laplacian regularization

Let the degree matrix  $D$  be a diagonal matrix where the  $k$ -th entry is evaluated as follows:

$$[D]_{kk} = \sum_{\ell=1}^N a_{k\ell}. \quad (3.7)$$

If we let

$$L = D - A \quad (3.8)$$

denote the Laplacian matrix, then the smoothness of the graph signal  $\widehat{\Gamma}_i$  can be measured in terms of the quadratic form of the graph Laplacian [2], [11]:

$$\mathcal{S}(\widehat{\Gamma}_i) = \left(\widehat{\Gamma}_i\right)^\top L \widehat{\Gamma}_i = \frac{1}{2} \sum_{(k,\ell) \in \mathcal{E}} a_{k\ell} (\widehat{\gamma}_k(i) - \widehat{\gamma}_\ell(i))^2. \quad (3.9)$$

The graph Laplacian regularizer (3.9) is suitable for applications where the smoothness of the signal with respect to the underlying graph must be promoted. The smaller  $\mathcal{S}(\widehat{\Gamma}_i)$  is, the smoother  $\widehat{\Gamma}_i$  on the graph is.

## 3.3 Regularized Optimization Problem

Motivated by the previous discussion, we propose to solve the following optimization problem at each agent  $k$ :

$$w_k^* \triangleq \underset{w_k}{\operatorname{argmin}} J_k(w_k) + r_k(w_k, \{w_\ell\}), \quad (3.10)$$

with

$$r_k(w_k, \{w_\ell\}) \triangleq \eta \sum_{\ell \in \mathcal{N}_k} a_{k\ell} \mathbb{E} f(\widehat{\gamma}_k(i) - \widehat{\gamma}_\ell(i)), \quad (3.11)$$

where the expectation is computed over the distribution of the random variables  $\{\widehat{\gamma}_k(i)\}$  and where  $\eta$  is a positive regularization parameter that ensures a tradeoff between the fidelity to the measurements and the prior information on the relationships between the labels. The function  $f: \mathbb{R} \rightarrow \mathbb{R}$  reduces to the absolute value function  $f(x) = |x|$  in sparsity promoting settings (3.6) and to the quadratic function  $f(x) = (x)^2$  in smoothness promoting settings (3.9). Since the function  $f(\cdot)$  can be non-differentiable (in the sparsity promoting setting), we employ the subgradient approach [27] to solve problem (3.10). For a mathematical tractability, and in order to be able to compute the subgradients in the optimization process, we replace the non-smooth  $\operatorname{sign}(\cdot)$  function in  $\widehat{\gamma}_k(i) = \operatorname{sign}(\mathbf{h}_{k,i}^\top w_k)$  by a smooth approximation given by the hyperbolic tangent function  $\tanh(\cdot)$  [28]. Therefore, instead of solving (3.10), agent  $k$  solves the following optimization problem:

$$\underset{w_k}{\operatorname{argmin}} J_k(w_k) + \widetilde{r}_k(w_k, \{w_\ell\}), \quad (3.12)$$

where

$$\tilde{r}_k(w_k, \{w_\ell\}) \triangleq \eta \sum_{\ell \in \mathcal{N}_k} a_{k\ell} \mathbb{E} f(\tanh(c\mathbf{h}_{k,i}^\top w_k) - \tanh(c\mathbf{h}_{\ell,i}^\top w_\ell)), \quad (3.13)$$

and where the scalar constant  $c$  controls the slope of the  $\tanh(\cdot)$  function. Figure 3.1 illustrates the plot of  $\text{sign}(x)$  along with the plots of  $\tanh(cx)$  for different values of  $c$ . It can be observed that as  $c \rightarrow \infty$ , the hyperbolic tangent approximation approaches the standard  $\text{sign}(\cdot)$  function.

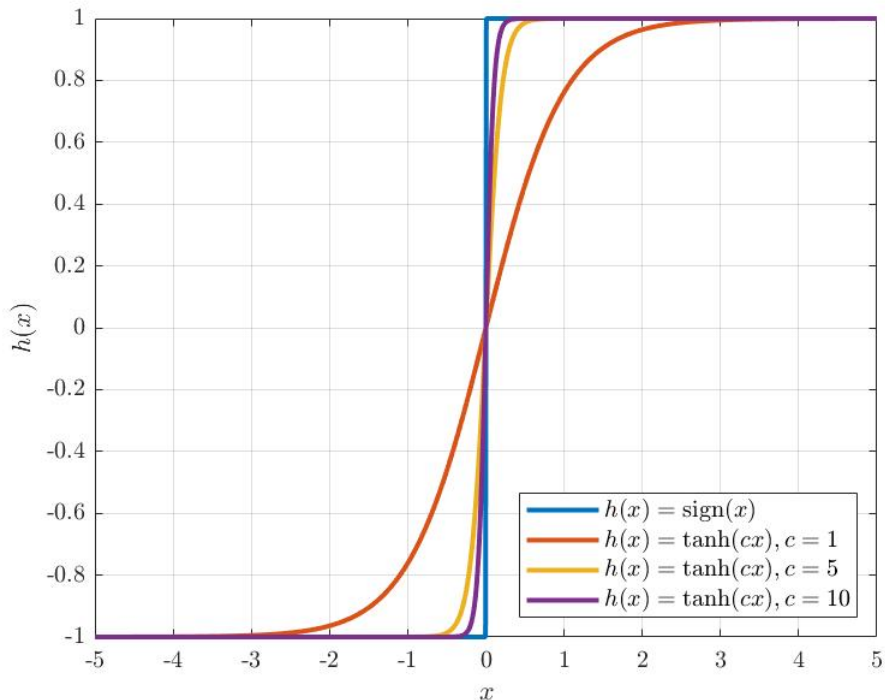


Figure 3.1: Sign function  $\text{sign}(x)$  and its hyperbolic tangent approximations  $\tanh(cx)$  for several values of  $c$  [1].

### 3.4 Decentralized Semi-Supervised Multitask Learning Algorithm [1]

Since the distributions of the random data  $\{\mathbf{h}_{k,i}, \gamma_k(i)\}$  are unknown, agent  $k$  will need to learn directly from the observed data samples by replacing the true gradient (and subgradient) vectors by their stochastic approximations. By using the gradient (and subgradient) vectors of the loss functions as an approximation, we arrive at Algorithm 1 for solving (3.12). Agent  $k$  starts with  $w_{k,-1}$ , an initial random guess of  $w_k^o$  in (3.3). The semi-supervised setting is considered through the binary variable  $\epsilon_k(i)$  that is equal to 1 if agent  $k$  is able to access its true label at iteration  $i$ , and 0 otherwise. At each iteration  $i$ , agent  $k$  collects the feature vector  $\mathbf{h}_{k,i}$ , and also collects the class label  $\gamma_k(i)$  when  $\epsilon_k(i) = 1$ .

At every iteration  $i$ , agent  $k$  performs two steps, which are demonstrated in Algorithm 1. The first step (3.15) is referred to as the *self-learning* step. We recall that, in the stochastic setting, the true gradient vector  $\nabla_{w_k} J_k(\cdot)$  of the risk  $J_k(\cdot)$  defined in (3.4) is unknown. Hence, agent  $k$  updates its estimate  $w_{k,i-1}$  in this step by stepping in the opposite direction of the stochastic approximation for the gradient vector of the risk  $J_k(\cdot)$ , scaled by a small positive step-size  $\mu$  [7]. We employ the following gradient approximation:

$$\widehat{\nabla_{w_k} J_k}(w_k) = -\frac{\gamma_k(i) \cdot \mathbf{h}_{k,i} \exp(-\gamma_k(i) \mathbf{h}_{k,i}^\top w_k)}{1 + \exp(-\gamma_k(i) \mathbf{h}_{k,i}^\top w_k)} + \rho w_k. \quad (3.14)$$

It should be noted that the variable  $\epsilon_k(i)$  is added to this step since the gradient approximation cannot be computed when the label is not observed (i.e., when  $\epsilon_k(i) = 0$ ).

---

**Algorithm 1:** Logistic semi-supervised learning over multitask graphs

---

Initialize:  $w_{k,-1}$  for every agent  $k$ ;

**for** every iteration  $i \geq 0$  **do**

**for** every agent  $k$  **do**

        collect the feature vector  $\mathbf{h}_{k,i}$ ;

        set  $\epsilon_k(i)$  to 1 if a label  $\gamma_k(i)$  is available, and to 0 otherwise;

$$\psi_{k,i} = w_{k,i-1} - \mu \epsilon_k(i) \widehat{\nabla_{w_k} J_k}(w_{k,i-1}) \quad (3.15)$$

$$w_{k,i} = \psi_{k,i} - \mu \widehat{\partial_{w_k} \tilde{r}_k}(\psi_{k,i}, \{\psi_{\ell,i}\}) \quad (3.16)$$

**end**

**end**

---

In the second step (3.16), which is referred to as the *social learning* step, agent  $k$  collaborates with its neighbors  $\ell \in \mathcal{N}_k$  by stepping in the opposite direction of the stochastic approximation of the subgradient (or gradient) vector of the regularizer  $\tilde{r}_k(\cdot, \{\psi_{\ell,i}\})$  defined in (3.13), also scaled by the step-size parameter  $\mu$ . In the sparsity promoting case (i.e., when  $f(x) = |x|$ ), we employ the chain rule in [29, p. 42] to evaluate the subdifferential of the regularization function  $\tilde{r}_k(w_k, \{w_\ell\})$ . The subdifferential is a generalized concept for unnecessarily differentiable functions. More specifically, the subdifferential of  $\tilde{r}_k(w_k, \{w_\ell\})$  at  $w_k$  is the set of all the possible subgradient vectors at  $w_k$ , which reduces to a single vector if the function is differentiable. Since, in the sparsity promoting case,  $\tilde{r}_k(w_k, \{w_\ell\})$  is not differentiable, the subgradients are not unique. Hence, we should select a single form to be adopted during the whole learning process. In our proposed algorithm, we choose to employ

the following subgradient approximation with respect to  $w_k$  given  $w_\ell$ :

$$\begin{aligned} \widehat{\partial_{w_k} \widetilde{r}_k}(w_k, w_\ell) = & \eta \sum_{\ell \in \mathcal{N}_k} a_{k\ell} c \mathbf{h}_{k,i} \left( 1 - (\tanh(c \mathbf{h}_{k,i}^\top w_k))^2 \right) \\ & \cdot \text{sign} \left( \tanh(c \mathbf{h}_{k,i}^\top w_k) - \tanh(c \mathbf{h}_{\ell,i}^\top w_\ell) \right), \end{aligned} \quad (3.17)$$

where the  $\text{sign}(\cdot)$  operator is evaluated as in (3.2). Moving now to the smoothness promoting case (i.e., when  $f(x) = (x)^2$ ), the regularization function becomes differentiable and the gradient approximation with respect to  $w_k$  given  $w_\ell$  is:

$$\begin{aligned} \widehat{\partial_{w_k} \widetilde{r}_k}(w_k, w_\ell) = & \eta \sum_{\ell \in \mathcal{N}_k} a_{k\ell} c \mathbf{h}_{k,i} \left( 1 - (\tanh(c \mathbf{h}_{k,i}^\top w_k))^2 \right) \\ & \cdot \left( \tanh(c \mathbf{h}_{k,i}^\top w_k) - \tanh(c \mathbf{h}_{\ell,i}^\top w_\ell) \right). \end{aligned} \quad (3.18)$$

According to (3.17) and (3.18), agent  $k$  needs to collaborate with its neighbors  $\ell \in \mathcal{N}_k$  by collecting from them the *scalar* values  $\{ \tanh(c \mathbf{h}_{\ell,i}^\top \boldsymbol{\psi}_{\ell,i}) \}$  to be able to perform step (3.16). These values can be interpreted as the intermediate predictions of the labels at neighboring agents and are interchanged as a message passing over the graph.

# CHAPTER 4

## STABILITY ANALYSIS

We now move to examine the stability of Algorithm 1 in the mean-square-error sense. This chapter uses the results derived in the third section of [1] and is dedicated to proving the following theorem.

**Theorem 1 (Network mean-square-error stability)** *Assuming that the components of the feature vectors  $\mathbf{h}_{k,i}$  cannot grow infinitely, and under Assumptions 1, 2, and 3, if the step-size parameter  $\mu$  satisfies:*

$$\mu < \frac{2 \min_{1 \leq k \leq N} q_k \lambda_k}{(\min_{1 \leq k \leq N} q_k \lambda_k)^2 + \beta_{q,\max}^2}, \quad (4.1)$$

then, in the limit, it holds that:

$$\limsup_{i \rightarrow \infty} \mathbb{E} \|\tilde{\mathbf{w}}_i\|^2 \leq \mathcal{O}(\mu) + \mathcal{O}(\mu\eta^2) + \mathcal{O}(\eta). \quad (4.2)$$

That is, for large  $i$ , and for a small enough step-size  $\mu$ , Algorithm 1 is stable and converges in the mean-square-error sense.

### 4.1 Network Error Vector Recursion

First, for each node  $k$ , we define the error vector  $\tilde{\mathbf{w}}_{k,i}$  as:

$$\tilde{\mathbf{w}}_{k,i} \triangleq w_k^\circ - \mathbf{w}_{k,i}, \quad (4.3)$$

and the intermediate error vector  $\tilde{\boldsymbol{\psi}}_{k,i}$  as:

$$\tilde{\boldsymbol{\psi}}_{k,i} \triangleq w_k^\circ - \boldsymbol{\psi}_{k,i}. \quad (4.4)$$

We also define the gradient noise vector  $\mathbf{s}_{k,i}(\cdot)$  as the difference between the true gradient and its approximation at iteration  $i$ , namely,

$$\mathbf{s}_{k,i}(w) = \nabla_{w_k} J_k(w) - \widehat{\nabla}_{w_k} J_k(w). \quad (4.5)$$

It is observed that the gradient approximation can be expressed as:

$$\widehat{\nabla}_{w_k} J_k(\mathbf{w}_{k,i-1}) = \nabla_{w_k} J_k(\mathbf{w}_{k,i-1}) - \mathbf{s}_{k,i}(\mathbf{w}_{k,i-1}). \quad (4.6)$$

Now, by using the mean-value theorem for real arguments [7], we can write:

$$\nabla_{w_k} J_k(\mathbf{w}_{k,i-1}) - \nabla_{w_k} J_k(w_k^\circ) = -\mathbf{H}_{k,i-1} \tilde{\mathbf{w}}_{k,i-1}, \quad (4.7)$$

where

$$\mathbf{H}_{k,i-1} \triangleq \int_0^1 \nabla_{w_k}^2 J_k(w_k^\circ - t\tilde{\mathbf{w}}_{k,i-1}) dt. \quad (4.8)$$

By combining (4.6) and (4.7), we can write the gradient approximation in (4.6) as follows:

$$\widehat{\nabla_{w_k} J_k}(\mathbf{w}_{k,i-1}) = -\mathbf{H}_{k,i-1} \tilde{\mathbf{w}}_{k,i-1} + \nabla_{w_k} J_k(w_k^\circ) - \mathbf{s}_{k,i}(\mathbf{w}_{k,i-1}). \quad (4.9)$$

Subtracting  $w_k^\circ$  from both sides of (3.15) and (3.16), and using (4.9), we obtain:

$$\tilde{\boldsymbol{\psi}}_{k,i} = (I_{M_k} - \mu\epsilon_k(i)\mathbf{H}_{k,i-1})\tilde{\mathbf{w}}_{k,i-1} + \mu\epsilon_k(i)\nabla_{w_k} J_k(w_k^\circ) - \mu\epsilon_k(i)\mathbf{s}_{k,i}(\mathbf{w}_{k,i-1}), \quad (4.10)$$

$$\tilde{\mathbf{w}}_{k,i} = \tilde{\boldsymbol{\psi}}_{k,i} + \mu\widehat{\partial_{w_k} r_k}(\boldsymbol{\psi}_{k,i}, \{\boldsymbol{\psi}_{\ell,i}\}). \quad (4.11)$$

Substituting (4.10) into (4.11), we get that the error vector for agent  $k$  evolves according to the following equation:

$$\begin{aligned} \tilde{\mathbf{w}}_{k,i} &= (I_{M_k} - \mu\epsilon_k(i)\mathbf{H}_{k,i-1})\tilde{\mathbf{w}}_{k,i-1} + \mu\epsilon_k(i)\nabla_{w_k} J_k(w_k^\circ) - \mu\epsilon_k(i)\mathbf{s}_{k,i}(\mathbf{w}_{k,i-1}) \\ &\quad + \mu\widehat{\partial_{w_k} r_k}(\boldsymbol{\psi}_{k,i}, \{\boldsymbol{\psi}_{\ell,i}\}). \end{aligned} \quad (4.12)$$

We now denote the network error vector as follows:

$$\tilde{\mathbf{w}}_i = \text{col}\{\tilde{\mathbf{w}}_{k,i}\}_{k=1}^N. \quad (4.13)$$

Using the notation (4.13), we construct from (4.12) the following recursion for the network error vector:

$$\tilde{\mathbf{w}}_i = \mathbf{B}_{i-1}\tilde{\mathbf{w}}_{i-1} + \mu\boldsymbol{\mathcal{E}}_i b - \mu\boldsymbol{\mathcal{E}}_i \mathbf{s}_i + \mu\mathbf{r}_i, \quad (4.14)$$

where

$$\boldsymbol{\mathcal{H}}_{i-1} \triangleq \text{diag}\{\mathbf{H}_{k,i-1}\}_{k=1}^N, \quad (4.15)$$

$$\mathbf{B}_{i-1} \triangleq I - \mu\boldsymbol{\mathcal{E}}_i \boldsymbol{\mathcal{H}}_{i-1}, \quad (4.16)$$

$$b \triangleq \text{col}\{\nabla_{w_k} J_k(w_k^\circ)\}_{k=1}^N, \quad (4.17)$$

$$\boldsymbol{\mathcal{E}}_i \triangleq \text{diag}\{\epsilon_k(i)I_{M_k}\}_{k=1}^N, \quad (4.18)$$

$$\mathbf{s}_i \triangleq \text{col}\{\mathbf{s}_{k,i}(\mathbf{w}_{k,i-1})\}_{k=1}^N, \quad (4.19)$$

$$\mathbf{r}_i \triangleq \text{col}\left\{\widehat{\partial_{w_k} r_k}(\boldsymbol{\psi}_{k,i}, \{\boldsymbol{\psi}_{\ell,i}\})\right\}_{k=1}^N. \quad (4.20)$$

## 4.2 Assumptions

Before proceeding, we introduce some assumptions that are helpful in examining the behavior of Algorithm 1. First, we introduce the following assumption on the variable  $\epsilon_k(i)$ .

**Assumption 1 (Semi-supervised modeling variable)** *The semi-supervised modeling variable  $\epsilon_k(i)$  is assumed to be Bernoulli randomly distributed, i.e.,*

$$\epsilon_k(i) = \begin{cases} 1, & \text{with probability } q_k \\ 0, & \text{with probability } 1 - q_k \end{cases}, \quad (4.21)$$

with  $0 < q_k \leq 1$ . It is also assumed that  $\epsilon_k(i)$  is independent of all the other random mechanisms in the networked system.

Furthermore, we introduce the following conditions on the twice differentiable costs  $\{J_k(w_k)\}$  and on the gradient noise processes  $\{\mathbf{s}_{k,i}\}$ , which are commonly assumed in the literature [7]. It can be shown that these conditions are satisfied by the logistic regression costs in (3.4) and by the gradient approximation (3.14).

**Assumption 2 (Strong convexity)** *The Hessian matrix function  $\nabla_{w_k}^2 J_k(w_k)$  is bounded from below and above as follows:*

$$0 < \lambda_{k,\min} I_{M_k} \leq \nabla_{w_k}^2 J_k(w_k) \leq \lambda_{k,\max} I_{M_k}, \quad (4.22)$$

where  $\lambda_{k,\min}$  and  $\lambda_{k,\max}$  are the smallest and largest eigenvalues of the Hessian matrix at agent  $k$ , respectively.

**Assumption 3 (Gradient noise process)** *The gradient noise process that is defined in (4.5) is assumed to satisfy the following conditions for  $1 \leq k \leq N$ :*

$$\mathbb{E}[\mathbf{s}_{k,i}(\mathbf{w}_k) \mid \mathcal{F}_{i-1}] = 0, \quad (4.23)$$

$$\mathbb{E}[\|\mathbf{s}_{k,i}(\mathbf{w}_k)\|^2 \mid \mathcal{F}_{i-1}] \leq \beta_k^2 \|\mathbf{w}_k\|^2 + \sigma_{s,k}^2, \quad (4.24)$$

for some  $\beta_k^2 \geq 0$ ,  $\sigma_{s,k}^2 \geq 0$ , and where  $\mathcal{F}_{i-1}$  denotes the collection of the past iterates  $\{\mathbf{w}_{k,j} \mid \forall k = 1, \dots, N \text{ and } j \leq i - 1\}$ .

## 4.3 Network Mean-Square-Error Recursion

Since  $w_k^\circ$  is the unique minimizer of the strongly convex function  $J_w(w_k)$ , the second term on the right-hand side of (4.14) is zero, and we can write:

$$\tilde{\mathbf{w}}_i = \mathbf{B}_{i-1} \tilde{\mathbf{w}}_{i-1} - \mu \mathbf{E}_i \mathbf{s}_i + \mu \mathbf{r}_i. \quad (4.25)$$

Since the variable  $\epsilon_k(i)$  modeling the semi-supervised setting follows a Bernoulli distribution, we have that:

$$\mathbb{E}(\epsilon_k(i))^2 = \mathbb{E}(\epsilon_k(i)) = q_k. \quad (4.26)$$



Squaring both sides of (4.25), computing the conditional expectation, and using Assumptions 1 and 3, we obtain the following relation:

$$\begin{aligned} \mathbb{E}[\|\tilde{\mathbf{w}}_i\|^2 \mid \mathcal{F}_{i-1}] &= \mathbb{E}[\|\mathcal{B}_{i-1}\tilde{\mathbf{w}}_{i-1}\|^2 \mid \mathcal{F}_{i-1}] + 2\mu\mathbb{E}[\tilde{\mathbf{w}}_{i-1}^\top \mathcal{B}_{i-1}^\top \mathbf{r}_i \mid \mathcal{F}_{i-1}] \\ &\quad + \mu^2\mathbb{E}[\|\mathcal{E}_i \mathbf{s}_i\|^2 \mid \mathcal{F}_{i-1}] + \mu^2\mathbb{E}[\|\mathbf{r}_i\|^2 \mid \mathcal{F}_{i-1}]. \end{aligned} \quad (4.27)$$

Let us consider the mean-square-error recursion (4.27) and let us first examine the third term on its right-hand side. By using Assumptions 1 and 3, the third term on the right-hand side of (4.27) can be bounded according to:

$$\begin{aligned} \mathbb{E}[\|\mathcal{E}_i \mathbf{s}_i\|^2 \mid \mathcal{F}_{i-1}] &= \sum_{k=1}^N \mathbb{E}[\|\boldsymbol{\epsilon}_k(i) \mathbf{s}_{k,i}\|^2 \mid \mathcal{F}_{i-1}] \\ &= \sum_{k=1}^N \mathbb{E}(\boldsymbol{\epsilon}_k(i))^2 \mathbb{E}[\|\mathbf{s}_{k,i}\|^2 \mid \mathcal{F}_{i-1}] \\ &\stackrel{(a)}{=} \sum_{k=1}^N q_k \mathbb{E}[\|\mathbf{s}_{k,i}\|^2 \mid \mathcal{F}_{i-1}] \\ &\stackrel{(b)}{\leq} \sum_{k=1}^N \left( q_k \bar{\beta}_k^2 \|\tilde{\mathbf{w}}_{k,i-1}\|^2 + q_k \bar{\sigma}_{s,k}^2 \right) \\ &\leq \left( \max_{1 \leq k \leq N} q_k \bar{\beta}_k^2 \right) \sum_{k=1}^N \|\tilde{\mathbf{w}}_{k,i-1}\|^2 + \sum_{k=1}^N q_k \bar{\sigma}_{s,k}^2, \end{aligned} \quad (4.28)$$

where in step (a) we use (4.26), while in (b) we extend condition (4.24) to the vector  $\tilde{\mathbf{w}}_{k,i-1}$  with  $\bar{\beta}_k^2$  and  $\bar{\sigma}_{s,k}^2$  denoting some positive scalar quantities. Hence, the third term on the right-hand side of (4.27) can be upper bounded by:

$$\mu^2 \mathbb{E}[\|\mathcal{E}_i \mathbf{s}_i\|^2 \mid \mathcal{F}_{i-1}] \leq \mu^2 \beta_{q,\max}^2 \|\tilde{\mathbf{w}}_{i-1}\|^2 + \mu^2 \sigma_{q,s}^2, \quad (4.29)$$

where

$$\beta_{q,\max}^2 \triangleq \max_{1 \leq k \leq N} q_k \bar{\beta}_k^2, \quad (4.30)$$

$$\sigma_{q,s}^2 \triangleq \sum_{k=1}^N q_k \bar{\sigma}_{s,k}^2. \quad (4.31)$$

Now, let us consider the fourth term on the right-hand side of (4.27). Assuming that the components of the feature vectors  $\mathbf{h}_{k,i}$  cannot grow infinitely, the partial sub-derivatives of the regularization functions  $\tilde{r}_k(w_k, \{w_\ell\})$  in (3.13) are bounded by some  $\zeta > 0$  for  $1 \leq k \leq N$  and for  $1 \leq m \leq M_k$  as follows:

$$\frac{1}{\eta} \left| \left[ \frac{\partial \tilde{r}_k}{\partial w_k}(\boldsymbol{\psi}_{k,i}, \{\boldsymbol{\psi}_{\ell,i}\}) \right]_m \right| \leq \zeta, \quad (4.32)$$

which implies that:

$$\mathbb{E}[\|\mathbf{r}_i\| \mid \mathcal{F}_{i-1}] \leq \eta\zeta \sqrt{\sum_{k=1}^N M_k}. \quad (4.33)$$

Therefore, we have the following upper-bound for the fourth term on the right-hand side of (4.27):

$$\mu^2 \mathbb{E}[\|\mathbf{r}_i\|^2 \mid \mathcal{F}_{i-1}] \leq \mu^2 \eta^2 \zeta^2 \sum_{k=1}^N M_k. \quad (4.34)$$

Let us now examine the first term on the right-hand side of (4.27). From Assumption 2, we have:

$$(1 - \mu\epsilon_k(i)\lambda_{k,\max})I_{M_k} \leq I_{M_k} - \mu\epsilon_k(i)\mathbf{H}_{k,i-1} \leq (1 - \mu\epsilon_k(i)\lambda_{k,\min})I_{M_k}. \quad (4.35)$$

For simplicity, we introduce the following approximation for the first term on the right-hand side of (4.27):

$$\begin{aligned} \mathbb{E}[\|\mathcal{B}_{i-1}\tilde{\mathbf{w}}_{i-1}\|^2 \mid \mathcal{F}_{i-1}] &\stackrel{(a)}{=} \tilde{\mathbf{w}}_{i-1}^\top (I - \mu\bar{\mathcal{E}}\mathcal{H}_{i-1} - \mu\mathcal{H}_{i-1}\bar{\mathcal{E}} + \mu^2\mathcal{H}_{i-1}\bar{\mathcal{E}}\mathcal{H}_{i-1})\tilde{\mathbf{w}}_{i-1} \\ &\stackrel{(b)}{\approx} \|(I - \mu\bar{\mathcal{E}}\mathcal{H}_{i-1})\tilde{\mathbf{w}}_{i-1}\|^2, \end{aligned} \quad (4.36)$$

where in step (a) we use the fact that:

$$\bar{\mathcal{E}} = \mathbb{E}\mathcal{E}_i = \mathbb{E}\mathcal{E}_i^2 = \text{diag}\{q_k I_{M_k}\}_{k=1}^N, \quad (4.37)$$

and in step (b) we use the fact that the step-size  $\mu$  is sufficiently small so that terms that depend on higher order powers of the step-size can be ignored. Combining (4.16) and (4.35), and using (4.36), we obtain the following upper-bound on the 2-induced norm of the block diagonal symmetric matrix  $\mathcal{B}'_{i-1} \triangleq I - \mu\bar{\mathcal{E}}\mathcal{H}_{i-1}$ :

$$\|\mathcal{B}'_{i-1}\| \leq \nu, \quad (4.38)$$

with

$$\nu \triangleq \max_{1 \leq k \leq N} \{\nu_k\}, \quad (4.39)$$

and where:

$$\nu_k \triangleq \max \left\{ |1 - \mu q_k \lambda_{k,\min}|, |1 - \mu q_k \lambda_{k,\max}| \right\}. \quad (4.40)$$

For simplicity, we write  $\nu$  in the following form:

$$\nu \triangleq \max_{1 \leq k \leq N} \left\{ |1 - \mu q_k \lambda_k| \right\}, \quad (4.41)$$

where:

$$\lambda_k = \begin{cases} \lambda_{k,\min}, & \text{if } |1 - \mu q_k \lambda_{k,\min}| \geq |1 - \mu q_k \lambda_{k,\max}| \\ \lambda_{k,\max}, & \text{otherwise} \end{cases}. \quad (4.42)$$

Using the sub-multiplicative property of the induced matrix norms, we arrive to the following inequality:

$$\begin{aligned}\|\mathbf{B}'_{i-1}\tilde{\mathbf{w}}_{i-1}\|^2 &\leq \|\mathbf{B}'_{i-1}\|^2\|\tilde{\mathbf{w}}_{i-1}\|^2 \\ &\leq \nu^2\|\tilde{\mathbf{w}}_{i-1}\|^2.\end{aligned}\tag{4.43}$$

Let us finally consider the second term on the right-hand side of (4.27). Let  $\eta\phi_{\max}$  be a bound on the largest component of  $\mathbf{B}'_{i-1}\mathbf{r}_i$  in absolute value for all  $i$ . Hence, the second term on the right-hand side of (4.27) can be upper-bounded by:

$$\begin{aligned}2\mu\mathbb{E}[\tilde{\mathbf{w}}_{i-1}^\top\mathbf{B}'_{i-1}\mathbf{r}_i \mid \mathcal{F}_{i-1}] &\leq 2\mu|\mathbb{E}[\tilde{\mathbf{w}}_{i-1}^\top\mathbf{B}'_{i-1}\mathbf{r}_i \mid \mathcal{F}_{i-1}]| \\ &\leq 2\mu\eta\phi_{\max} \cdot \|\mathbb{E}[\tilde{\mathbf{w}}_{i-1}]\|_1.\end{aligned}\tag{4.44}$$

## 4.4 Network Mean Error Recursion

To proceed with our analysis, we should examine the behavior of the mean error vector to find an upper-bound for the term in (4.44). Taking expectations of both sides in (4.25), and using condition (4.23), we arrive at the following recursion for the mean error vector:

$$\mathbb{E}[\tilde{\mathbf{w}}_i] = \mathbb{E}[\mathbf{B}_{i-1}\tilde{\mathbf{w}}_{i-1}] + \mu\mathbb{E}[\mathbf{r}_i].\tag{4.45}$$

Using notation (4.16), and using the fact that  $\bar{\mathcal{E}} = \mathbb{E}\mathcal{E}_i$ , we get:

$$\mathbb{E}[\tilde{\mathbf{w}}_i] = (I - \mu\bar{\mathcal{E}}\mathcal{H}_{i-1})\mathbb{E}[\tilde{\mathbf{w}}_{i-1}] + \mu\mathbb{E}[\mathbf{r}_i],\tag{4.46}$$

which can be alternatively written as:

$$\mathbb{E}[\tilde{\mathbf{w}}_i] = \mathbf{B}'_{i-1}\mathbb{E}[\tilde{\mathbf{w}}_{i-1}] + \mu\mathbb{E}[\mathbf{r}_i].\tag{4.47}$$

Iterating (4.47) starting from  $i = 1$ , we get:

$$\mathbb{E}[\tilde{\mathbf{w}}_i] = \prod_{j=0}^{i-1}\mathbf{B}'_j\mathbb{E}[\tilde{\mathbf{w}}_0] + \mu\sum_{j=1}^{i-1}\left(\mathbb{E}[\mathbf{r}_j]\prod_{k=j}^{i-1}\mathbf{B}'_k\right) + \mu\mathbb{E}[\mathbf{r}_i].\tag{4.48}$$

The first term on the right-hand side of (4.48) converges to zero as  $i \rightarrow \infty$  if the matrices  $\mathbf{B}'_j$  are stable, which is ensured by a small enough step-size  $\mu$ . Additionally, the series  $\mu\sum_{j=1}^{i-1}\left(\mathbb{E}[\mathbf{r}_j]\prod_{k=j}^{i-1}\mathbf{B}'_k\right)$  converge as  $i \rightarrow \infty$  since  $\mathbf{B}'_k$  are stable and  $\mathbb{E}[\mathbf{r}_j]$  are bounded according to (4.32). The third term is also bounded according to (4.32). Hence, for a sufficiently small  $\mu$ , and as  $i \rightarrow \infty$ , the mean error vector converges to a small region of the order of  $\mu\eta$ , i.e.:

$$\limsup_{i \rightarrow \infty}\mathbb{E}[\tilde{\mathbf{w}}_i] \leq \mathcal{O}(\mu\eta).\tag{4.49}$$

## 4.5 Network Mean-Square-Error Stability

Since the mean error vector  $\mathbb{E}[\tilde{\mathbf{w}}_{i-1}]$  converges to a small region as  $i \rightarrow \infty$  for a sufficiently small  $\mu$ ,  $\|\mathbb{E}[\tilde{\mathbf{w}}_{i-1}]\|_1$  can be upper bounded by some constant  $\kappa > 0$ . Hence, we can rewrite (4.44) as follows:

$$2\mu\mathbb{E}[\tilde{\mathbf{w}}_{i-1}^\top \mathbf{B}_{i-1}^\top \mathbf{r}_i \mid \mathcal{F}_{i-1}] \leq 2\mu\eta\phi_{\max}\kappa. \quad (4.50)$$

Substituting (4.29), (4.34), (4.43), and (4.50) in (4.27), and taking expectations again to eliminate the conditioning on  $\mathcal{F}_{i-1}$ , we arrive at:

$$\mathbb{E}\|\tilde{\mathbf{w}}_i\|^2 \leq \left(\nu^2 + \mu^2\beta_{q,\max}^2\right)\mathbb{E}\|\tilde{\mathbf{w}}_{i-1}\|^2 + \mu^2\left(\sigma_{q,s}^2 + \eta^2\zeta^2\sum_{k=1}^N M_k\right) + 2\mu\eta\phi_{\max}\kappa. \quad (4.51)$$

To simplify the notations, we introduce the following scalars:

$$\alpha \triangleq \nu^2 + \mu^2\beta_{q,\max}^2, \quad (4.52)$$

$$\tau \triangleq \mu^2\left(\sigma_{q,s}^2 + \eta^2\zeta^2\sum_{k=1}^N M_k\right) + 2\mu\eta\phi_{\max}\kappa, \quad (4.53)$$

and hence, we can rewrite (4.51) more compactly as:

$$\mathbb{E}\|\tilde{\mathbf{w}}_i\|^2 \leq \alpha\mathbb{E}\|\tilde{\mathbf{w}}_{i-1}\|^2 + \tau. \quad (4.54)$$

Iterating (4.54) starting from  $i = 1$ , we get:

$$\mathbb{E}\|\tilde{\mathbf{w}}_i\|^2 \leq \alpha^i\mathbb{E}\|\tilde{\mathbf{w}}_0\|^2 + \sum_{j=0}^{i-1} \alpha^j\tau, \quad (4.55)$$

which can be alternatively written as:

$$\mathbb{E}\|\tilde{\mathbf{w}}_i\|^2 \leq \alpha^i\mathbb{E}\|\tilde{\mathbf{w}}_0\|^2 + \tau\frac{1-\alpha^i}{1-\alpha}. \quad (4.56)$$

Taking the limit as  $i \rightarrow \infty$ , and if  $0 < \alpha < 1$ , we get:

$$\limsup_{i \rightarrow \infty} \mathbb{E}\|\tilde{\mathbf{w}}_i\|^2 \leq \frac{\tau}{1-\alpha}. \quad (4.57)$$

Since, from Assumption 1, we have that  $0 < q_k \leq 1$  for  $1 \leq k \leq N$ , a sufficiently small step-size  $\mu \ll 1$  ensures that:

$$\nu^2 = 1 - \mathcal{O}(\mu), \quad (4.58)$$

and therefore,

$$1 - \alpha = \mathcal{O}(\mu). \quad (4.59)$$

Hence, the upper-bound in (4.57) can be rewritten as follows:

$$\begin{aligned}
\limsup_{i \rightarrow \infty} \mathbb{E} \|\tilde{\mathbf{w}}_i\|^2 &\leq \frac{\tau}{1 - \alpha} \\
&\leq \frac{\mathcal{O}(\mu^2) + \mathcal{O}(\mu^2 \eta^2) + \mathcal{O}(\mu \eta)}{\mathcal{O}(\mu)} \\
&\leq \mathcal{O}(\mu) + \mathcal{O}(\mu \eta^2) + \mathcal{O}(\eta).
\end{aligned} \tag{4.60}$$

To ensure that  $0 < \alpha < 1$ , we need:

$$0 < \nu^2 + \mu^2 \beta_{q,\max}^2 < 1, \tag{4.61}$$

$$\max_{1 \leq k \leq N} \left\{ (1 - \mu q_k \lambda_k)^2 \right\} + \mu^2 \beta_{q,\max}^2 < 1. \tag{4.62}$$

A small enough step-size  $\mu \ll 1$  also ensures that  $0 < \nu^2 < 1$ , therefore we get:

$$(1 - \mu \min_{1 \leq k \leq N} q_k \lambda_k)^2 + \mu^2 \beta_{q,\max}^2 < 1, \tag{4.63}$$

$$\mu^2 (\min_{1 \leq k \leq N} q_k \lambda_k)^2 - 2\mu \min_{1 \leq k \leq N} q_k \lambda_k + \mu^2 \beta_{q,\max}^2 < 0, \tag{4.64}$$

$$\mu \left( (\min_{1 \leq k \leq N} q_k \lambda_k)^2 + \beta_{q,\max}^2 \right) < 2 \min_{1 \leq k \leq N} q_k \lambda_k. \tag{4.65}$$

Therefore, we arrive at condition (4.1) on the step-size  $\mu$  to ensure that  $0 < \alpha < 1$  and to achieve the upper-bound in (4.60), and hence finally arrive at Theorem 1 that illustrates the network behavior in the mean-square-error sense when considering the proposed Algorithm 1.

# CHAPTER 5

## EXPERIMENTAL RESULTS

To illustrate the effectiveness of the proposed decentralized semi-supervised multi-task learning algorithm, several experiments were conducted on synthetic and real data, and their results are presented and discussed in this chapter. Those results appear in the fourth section of [1].

### 5.1 Synthetic Data Experiments

Two synthetic data experiments were conducted by creating a network using the signal processing on graphs toolbox [30]. The network structure was generated according to the Stochastic Block Model (SBM) [31]. The graph parameters were chosen such that the network consists of  $N = 50$  nodes, divided equally into two clusters. The probability of connection between nodes in the same cluster was set to 0.2, while the probability of inter-cluster connection was 0.01. The edge weights are random numbers, where intra-cluster edges weights follow a normal distribution  $\mathcal{N}(2, 0.2)$  while inter-cluster edges weights are small random numbers between 0 and 1. The resulting network is illustrated in Figure 5.1. The difference between the two conducted experiments is in the selection of the agents that are able to observe their true labels. One experiment was conducted by fixing the informed nodes at all the time instants, while in the other experiment, random sampling was adopted similarly to [32]. At every time instant or iteration  $i$ , the true labels  $\gamma_k(i)$  were generated by a random choice between 1 and -1 for agents in the first cluster, and the opposite label was assigned for agents in the second cluster. Agents were observing different numbers of features, where the number of features  $M_k$  of agent  $k$  was randomly chosen from a discrete set of values ranging between 1 and 5. The  $m$ -th entry of the feature vector  $\mathbf{h}_{k,i}$  was generated according to the following equation:

$$[\mathbf{h}_{k,i}]_m = \gamma_k(i) \cdot \mathbf{e}_k(i) + \mathbf{v}_k(i), \quad (5.1)$$

where  $\mathbf{e}_k(i)$  and  $\mathbf{v}_k(i)$  are randomly generated from the Gaussian distributions  $\mathcal{N}(m, 0.5)$  and  $\mathcal{N}(0, 1)$ , respectively. The quantity  $\mathbf{v}_k(i)$  is intended to represent the zero-mean additive noise affecting the sensors of agent  $k$  at iteration  $i$ . After some hyper-parameter tuning, we set  $\mu$ ,  $c$ , and  $\rho$  equal to 0.01, 10, and 0.05, respec-

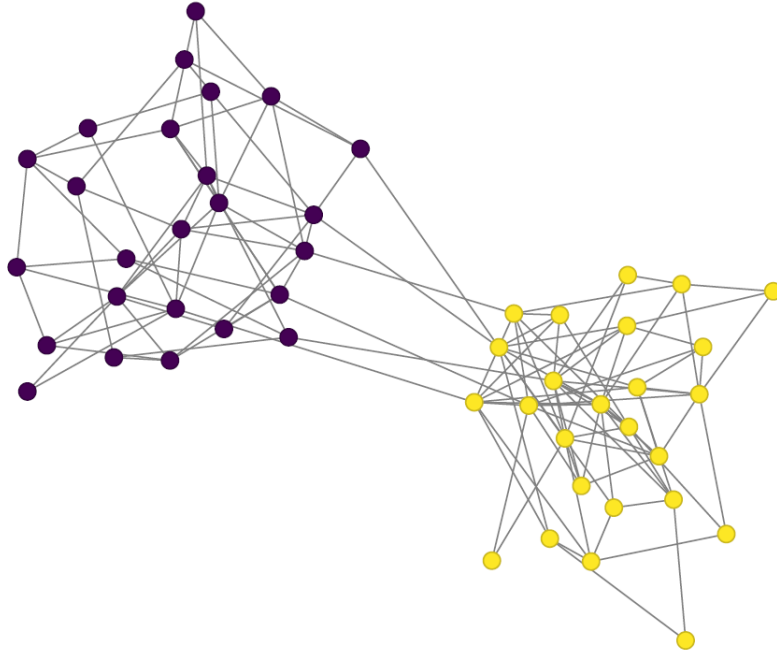


Figure 5.1: Clustered network structure (agents with the same color observe the same label) [1].

tively. The parameter vector  $w_k$  was initialized to a vector of zeros at every agent  $k$  before starting to run Algorithm 1.

Every agent  $k$  was tested on a separate validation set of  $J = 100$  samples at every iteration  $i$ . The validation loss is calculated as follows:

$$\text{VL}(i) = \frac{1}{2N} \sum_{k=1}^N \left( \frac{1}{J} \sum_{j=1}^J |\gamma_k(j) - \text{sign}(\mathbf{h}_{k,j}^\top \mathbf{w}_{k,i})| \right). \quad (5.2)$$

At last, we distinguish between the performances of the non-cooperative and cooperative scenarios with different choices of  $\eta$  based on the validation loss metric.

### 5.1.1 Experiment 1

This experiment was conducted by assuming that, within each cluster, only one node is observing a label. This results in two informed nodes in the whole network, and these informed nodes were fixed at every iteration. In other words, the probability  $q_k$  of these two nodes is set to 1, while the probability of the remaining nodes is set to 0. It should be noted that this setting was not studied in Chapter 4 since it was assumed that  $0 < q_k \leq 1$ , however, we include the results of this experiment to illustrate the findings in such scenario. Figures 5.2 and 5.3 report the results in terms of validation loss for 500 iterations. These results were averaged over 20 Monte-Carlo runs. Figure 5.2 pertains to the obtained results using the network Lasso regularizer that is used to promote sparsity, whereas Figure 5.3 illustrates

the results of smoothness promoting using the graph Laplacian regularizer. Three different values of  $\eta$  were considered: 0, illustrating the non-cooperative scenario, 0.001 and 0.01.

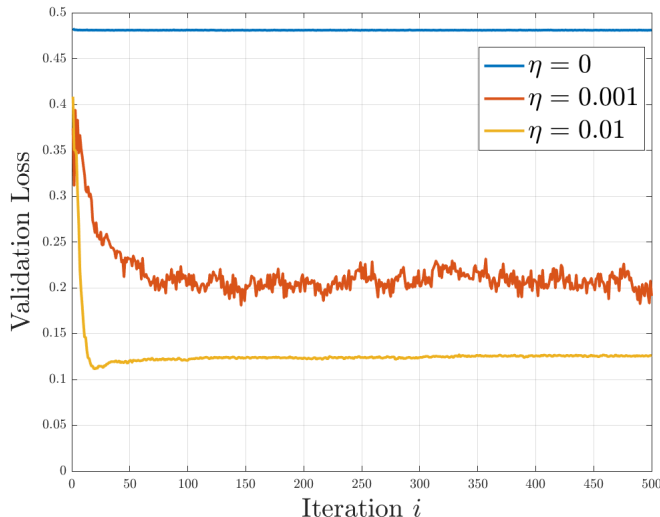


Figure 5.2: Synthetic data experiment 1: Network average validation loss while fixing the informed nodes and using the network Lasso regularization promoting sparsity ( $f(x) = |x|$ ) [1].

The results for both  $\ell_1$ -norm and squared  $\ell_2$ -norm settings indicate that for  $\eta = 0$  (non-cooperative case), the validation loss remains constant, approximately 0.48, during the whole learning process. Such value is considerably high since it is close to 0.5, which can be achieved by a random guess. This is possibly because the nodes with probability  $q_k = 0$  were not able to update their estimates in the first step (3.15) since their semi-supervised modeling variable  $\epsilon_k(i)$  is always equal to 0, and neither able to perform the second update step (3.16) since  $\eta$  is equal to 0. Hence, most of the network nodes kept their parameter vectors equal to the initial random guess  $w_{k,-1}$ .

On the other hand, when the network agents collaborate with each other by setting  $\eta > 0$ , the network average validation loss considerably decreases in both  $\ell_1$ -norm and squared  $\ell_2$ -norm settings. When using the sparsity promoting regularizer (Figure 5.2), the loss reaches around 0.2 and 0.12 for  $\eta = 0.001$  and  $\eta = 0.01$ , respectively. Additionally, in Figure 5.3, i.e., when promoting the graph signal smoothness, the loss reaches around 0.05 and 0.1 for  $\eta = 0.001$  and  $\eta = 0.01$ , respectively. Therefore, this suggests that cooperation is useful in such scenarios to help decrease the network classifiers' testing losses. Those results are promising since, in various applications, data labeling is not always possible due to several reasons: expensiveness, human assistance requirement, and privacy concerns.

### 5.1.2 Experiment 2

This experiment was conducted by assuming that each agent in the network is observing its true label with a probability  $q_k = 0.1 \forall k$ . The values of  $\eta$  that were



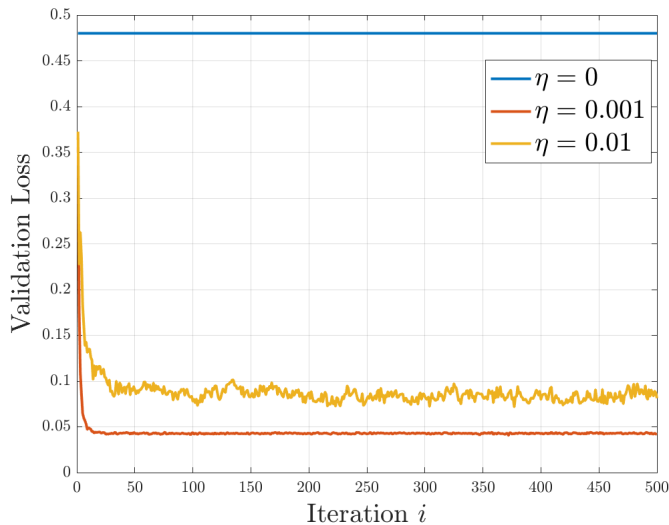


Figure 5.3: Synthetic data experiment 1: Network average validation loss while fixing the informed nodes and using the graph Laplacian regularization promoting smoothness ( $f(x) = x^2$ ) [1].

considered are 0, 0.001, and 0.005. The results are shown in Figures 5.4 and 5.5 for 500 iterations and are averaged over 20 Monte-Carlo runs. Similarly, Figure 5.4 represents the obtained results when using the network Lasso regularizer, while Figure 5.5 demonstrates the results when the graph Laplacian regularizer is used.

In this random sampling setting, no significant difference is observed between the steady-state validation losses of the non-cooperative ( $\eta = 0$ ) and the cooperative ( $\eta = \{0.001, 0.005\}$ ) implementations. In all these implementations, and for both  $\ell_1$ -norm and squared  $\ell_2$ -norm settings, the network validation loss converges to approximately 0.05. The reason why the loss reached in the non-cooperative case in this experiment (0.05) is lower than the loss of the non-cooperative case of the first experiment (0.48) could be that all the network agents in the random sampling setting were given a chance to update their parameter vectors in the first step (3.15). Although the final cooperative and non-cooperative losses are identical (for both sparsity and smoothness promoting settings), one key observation in this experiment is that cooperation improves the network convergence rate. For example, as illustrated in Figure 5.4, the network converges after approximately 10 iterations when  $\eta = \{0.001, 0.005\}$ , while more than 70 iterations are required to converge in the non-cooperative case. A similar behavior is also observed in Figure 5.5 when using the smoothness promoting regularizer. Such finding is encouraging when the data are limited to a small number of samples, for instance, in medical applications with insufficient amount of data from patients and privacy restrictions to share them.

## 5.2 Real Data Experiments

After checking the effectiveness of our proposed algorithm on synthetic data, we proceed to validate its efficacy in real applications. Hence, we chose to test Algorithm 1

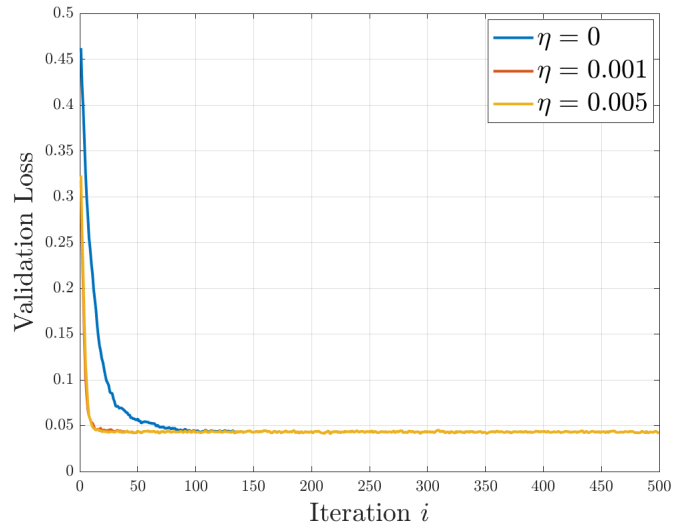


Figure 5.4: Synthetic data experiment 2: Network average validation loss while adopting the random sampling and using the network Lasso regularization promoting sparsity ( $f(x) = |x|$ ) [1].

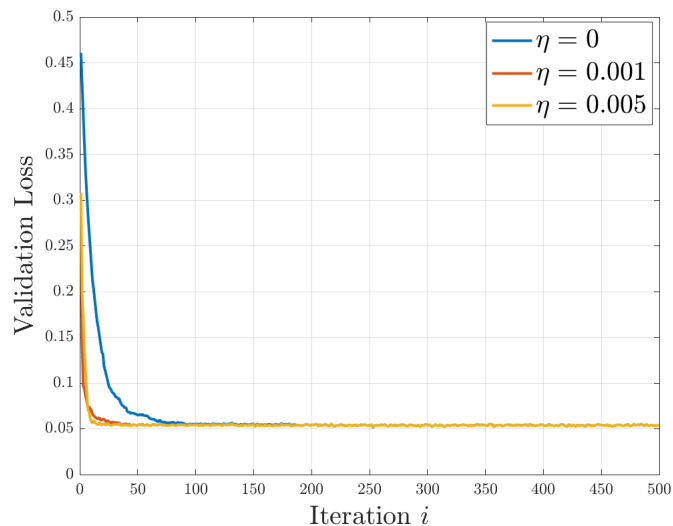


Figure 5.5: Synthetic data experiment 2: Network average validation loss while adopting the random sampling and using the graph Laplacian regularization promoting smoothness ( $f(x) = x^2$ ) [1].

on a weather dataset, the Global Historical Climatology Network - Monthly (GHCN-M) dataset [33], [34]. This dataset contains historical climate measurements from a large number of weather stations spread across the world. In our experiments, we limit our study to the measurements that were previously considered in [11], i.e., that belong to 139 stations located in the United States of America (USA) and that were collected during the period ranging from 2004 to 2017. The aim is to predict the rain (or snow) occurrence in each station based on five daily collected measurements, which are the features in this classification problem and consist of:

- Mean temperature
- Mean dew point
- Mean visibility
- Mean wind speed
- Maximum sustained wind speed

In order to run our simulations, we use the code of the experiments conducted in [11] to collect the daily measurements and to construct an undirected weighted graph  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, A\}$ , where each node represents a station. Hence, the total number of nodes is  $N = |\mathcal{V}| = 139$ . Let  $\mathcal{N}_{k,0}$  define the set of the 4-nearest neighbors of node  $k$ , which are evaluated based on the geographical distance between nodes or stations. The edges' weights of the adjacency matrix  $A$  are generated according to:

$$a_{k\ell} = \frac{p_{k\ell} + p_{\ell k}}{2}, \quad (5.3)$$

where the value  $p_{k\ell}$  is calculated as in [35]:

$$p_{k\ell} = \frac{\exp(-d_{k\ell}^2)}{\sqrt{\sum_{m \in \mathcal{N}_{k,0}} \exp(-d_{km}^2) \sum_{n \in \mathcal{N}_{\ell,0}} \exp(-d_{\ell n}^2)}}, \quad \ell \in \mathcal{N}_{k,0}, \quad (5.4)$$

where  $d_{k\ell}$  is the geodesic distance between the nodes  $k$  and  $\ell$ , which is defined as the number of edges of the shortest path between  $k$  and  $\ell$ . Similarly to the synthetic data experiments, the feature vector at agent  $k$  and iteration  $i$  is denoted by  $\mathbf{h}_{k,i}$ , which is a vector of size  $M = 5$  containing the five measurements at station  $k$  and day  $i$ . The label  $\gamma_k(i)$  is equal to 1 if rain or snow happened at station  $k$  and day  $i$ , and  $-1$  otherwise.

In our experiments, the dataset is split into training and testing sets in the same way as in [11]. The data samples from 2004 till 2012 were taken for training, i.e., for estimating the parameter vector  $w_k^o$  of each node or classifier  $k$ , resulting in 3288 training instances (3288 days). The testing set, which is intended to measure the accuracy of these classifiers, includes the remaining data (2012 till 2017), resulting in  $J = 1826$  testing samples (1826 days). As in [11], the first estimate of the parameter vectors was generated using the standard normal distribution before starting to run Algorithm 1 on the training set. This first estimate was fixed for all the experiments

that we run for different choices of  $\eta$  for a fair comparison. Heuristically, we set  $\mu$ ,  $c$ , and  $\rho$  equal to  $3 \cdot 10^{-4}$ , 10, and  $2 \cdot 10^{-5}$ , respectively. Similarly to the previous section, we also conducted two experiments by fixing the informed nodes in the first one and adopting random sampling in the second one. Finally, to compare between the non-cooperative and cooperative scenarios, the network performance is evaluated on the testing set by calculating the testing error as:

$$\frac{1}{2N} \sum_{k=1}^N \left( \frac{1}{J} \sum_{j=1}^J |\gamma_k(j) - \text{sign}(\mathbf{h}_{k,j}^\top \mathbf{w}_{k,\infty})| \right), \quad (5.5)$$

which is defined in a similar manner to (5.2), but the difference is that  $\mathbf{w}_{k,\infty}$  refers to the average of the last 200 estimates of the parameter vector at node  $k$ , as in [11].

### 5.2.1 Experiment 1

This experiment was conducted by fixing seven informed nodes (with probability  $q_k = 1$ ) at all the training days while all the remaining network nodes continuously have a probability  $q_k = 0$ . Figure 5.6 illustrates the constructed network of weather stations with the black nodes representing the informed nodes in the network. Only these seven informed nodes were able to access their labels and update the estimates of their parameter vectors at the first step of Algorithm 1. Table 5.1 lists the network average testing losses for several values of the regularization strength  $\eta$  when using the network Lasso regularizer (promoting sparsity). Additionally, Table 5.2 reports the results when using the graph Laplacian regularizer (promoting smoothness).

Table 5.1: Real data experiment 1: Network average testing error while fixing the informed nodes and using the network Lasso regularization promoting sparsity ( $f(x) = |x|$ ).

Regularization strength $\eta$	0	0.01	0.1	0.5	1	10	50
Testing error	0.4891	0.4675	0.4271	<b>0.3919</b>	0.3933	0.3942	0.423

Table 5.2: Real data experiment 1: Network average testing error while fixing the informed nodes and using the graph Laplacian regularization promoting smoothness ( $f(x) = x^2$ ).

Regularization strength $\eta$	0	0.01	0.1	1	10	20	50
Testing error	0.4891	<b>0.3897</b>	0.4013	0.3934	0.3944	0.3955	0.4003

The results, for both sparsity and smoothness promoting settings, show that the testing error in the non-cooperative scenario ( $\eta = 0$ ) is somewhat high (0.4891). This

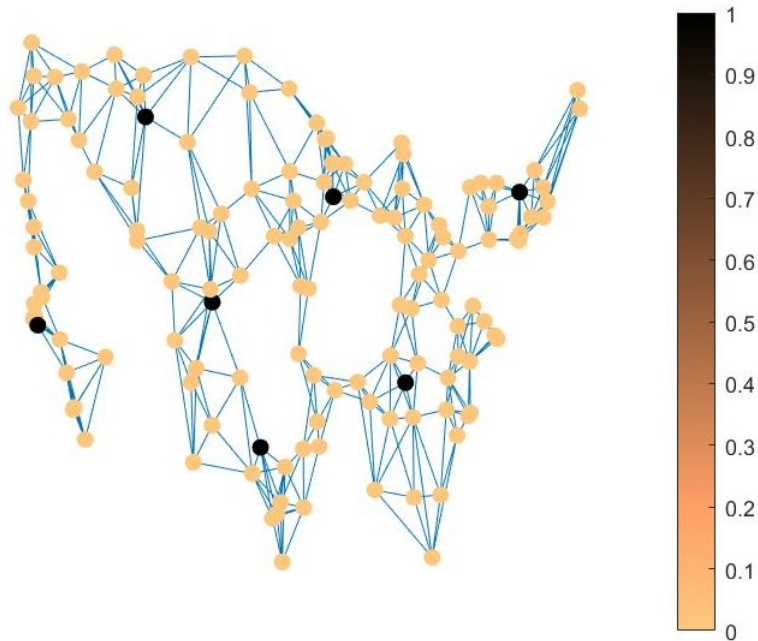
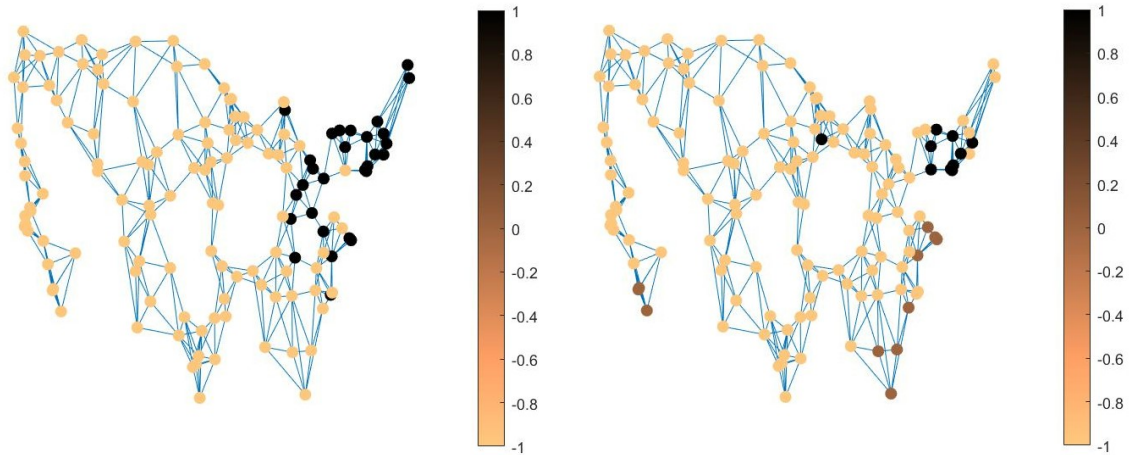


Figure 5.6: Illustration of the 139 weather stations (or network nodes) with the informed nodes painted in black (with probability  $q_k = 1$ ) and spread across the network.

is because uninformed nodes could not properly estimate their parameter vectors since their true labels were not available during the whole training phase, similarly to the first synthetic data experiment. On the other hand, by allowing the agents to collaborate ( $\eta > 0$ ), the testing error decreases in the two considered settings: It reaches its smallest value (0.3919) for  $\eta = 0.5$  in the  $\ell_1$ -norm setting, while the best performance in the squared  $\ell_2$ -norm setting is achieved for  $\eta = 0.01$  (0.3897). This demonstrates that the proposed Algorithm 1 has led to a decrease of the testing loss in this classification problem. It is worth noting that a higher value of  $\eta$  resulted in a slightly higher testing loss in both considered settings. This increase in bias could be the result of giving more importance to the regularization term than the logistic loss.

In Figure 5.7, we illustrate the labels of the 139 network nodes on March 13, 2014, with Figure 5.7a representing the real labels and Figure 5.7b pertaining to the predicted labels. The black color represents the positive class (rain or snow occurrence), while the copper color belongs to the negative class. The additional brown color in Figure 5.7b belongs to the predictions where  $\hat{\gamma}_k(i) = \text{sign}(\mathbf{h}_{k,i}^\top \mathbf{w}_k^o) = 0$ . In such cases, both 1 and  $-1$  values for the predicted labels can be accepted since both predictions are of equal probabilities. The average network testing loss reached on this specific day is 0.1871, which can be observed from the graphs since most of the nodes predictions agree with their real labels. One key observation is that the real labels in Figure 5.7a approve that neighboring classifiers in some applications are most likely to observe data samples belonging to the same class.

Although the results of this experiment validate the results of the first synthetic



(a) Real labels. The structure illustrates that nearby classifiers are more likely to observe similar labels.

(b) Predicted labels. The extra brown color represents the predictions where  $\hat{\gamma}_k(i) = \text{sign}(\mathbf{h}_{k,i}^\top \mathbf{w}_k^o) = 0$ .

Figure 5.7: Real data experiment 1: Illustration of the 139 weather stations along with their class labels on March 13, 2014. The black color corresponds to the label 1 (rainy or snowy day) whereas the copper color represents the label  $-1$  (non-rainy and non-snowy day).

data experiment, the error decrease in the latter is more noticeable than the decrease in this real data experiment. This could be the result of the imperfect clustering of the data in the real scenario. In other words, by comparing Figure 5.1 to Figure 5.7a, it is observable that the perfect data clustering that was created by the SBM is not always encountered in real applications, where the presence of outliers affects the clustered structure of the data. Additionally, in this experiment, all the network agents had access to five measurements, therefore, none of them was suffering from a lack of features in comparison to others. Accordingly, those facts might be limiting the performance of our proposed Algorithm 1 in this real data experiment.

### 5.2.2 Experiment 2

Similarly to the synthetic data experiment 2, each network agent in this experiment was also collecting its label with a probability  $q_k = 0.1 \forall k$ . The results that were obtained using the network Lasso regularizer are listed in Table 5.3 and the results when the graph Laplacian regularizer is used are summarized in Table 5.4.

In this experiment, the smallest testing loss is also achieved in a cooperative setting. It is equal to 0.3798 ( $\eta = 1$ ) in the  $\ell_1$ -norm setting, and equal to 0.3752 ( $\eta = 0.01$ ) in the squared  $\ell_2$ -norm setting. However, the difference between these values and the testing errors reached in the non-cooperative case is slight. These results are in accordance with the findings obtained in the synthetic data experiment 2 (subsection 5.1.2) since no significant difference existed between the final validation losses of the non-cooperative and cooperative scenarios.

Table 5.3: Real data experiment 2: Network average testing error while adopting the random sampling and using the network Lasso regularization promoting sparsity ( $f(x) = |x|$ ).

Regularization strength $\eta$	0	0.01	0.1	0.5	1	10	50
Testing error	0.3811	0.3811	0.3802	0.3827	<b>0.3798</b>	0.3827	0.3852

Table 5.4: Real data experiment 2: Network average testing error while adopting the random sampling and using the graph Laplacian regularization promoting smoothness ( $f(x) = x^2$ ).

Regularization strength $\eta$	0	0.01	0.1	1	10	20	50
Testing error	0.3904	<b>0.3752</b>	0.3808	0.3822	0.3822	0.3855	0.3822

# CHAPTER 6

## CONCLUSION

In this final chapter, we highlight the main takeaways of this thesis and then provide some future research directions.

### 6.1 Conclusion

In this study, we tackled a network semi-supervised online binary classification problem in a heterogeneous setting where classifiers may observe different numbers and types of features. We considered a prior knowledge that neighboring classifiers are more likely to observe data samples belonging to the same class, which is commonly encountered in many practical scenarios. To encourage the similarities between the labels of neighboring agents, we proposed to add a graph regularization term to the logistic loss function. Two additional regularization terms were proposed: the network Lasso regularization (promoting sparsity) represented by the  $\ell_1$ -norms of the differences between the labels and the graph Laplacian regularization (promoting smoothness) represented by the squared  $\ell_2$ -norms of the differences between the labels.

The proposed regularized problem was solved using the stochastic (sub-)gradient descent approach since the data were received in an online streaming manner. This resulted in an algorithm of two steps: a *self-learning* step and a *social learning* step. This algorithm's stability was studied in the mean-square-error sense, and assumptions and conditions were derived to ensure its convergence. The synthetic and real data experimental results indicate that, when fixing the nodes that are observing their true labels, the cooperation between neighboring agents helps in decreasing the validation and testing errors. Furthermore, the results obtained when randomly selecting the informed nodes demonstrate that the collaboration among agents mainly helps in accelerating the convergence speed of the algorithm.

### 6.2 Future Research Work

Two directions for future research may be investigated to possibly improve this work. The first one is to try to minimize the network Lasso regularization function differently, by employing the proximal operator for instance, since the subgradient



descent method might not necessarily be a descent method. The second future research recommendation is to consider variations of the hyperbolic tangent function  $\tanh(\cdot)$ , as in [36] for instance, because the  $\tanh(\cdot)$  function might sometimes suffer from the vanishing gradient problem [36]–[38].

# BIBLIOGRAPHY

- [1] M. Issa, R. Nassif, E. Rizk, and A. H. Sayed, “Decentralized semi-supervised learning over multitask graphs,” in *2022 56th Asilomar Conference on Signals, Systems, and Computers*, October 2022.
- [2] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” *IEEE Sig. Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.
- [3] A. Sayed, S.-Y. Tu, J. Chen, X. Zhao, and Z. Towfic, “Diffusion strategies for adaptation and learning over networks: An examination of distributed strategies and network behavior,” *IEEE Signal Processing Magazine*, vol. 30, pp. 155–171, May 2013.
- [4] R. Nassif, S. Vlaski, C. Richard, J. Chen, and A. H. Sayed, “Multitask learning over graphs: An approach for distributed, streaming machine learning,” *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 14–25, 2020.
- [5] R. Nassif, S. Vlaski, and A. H. Sayed, “Distributed inference over multitask graphs under smoothness,” in *2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, 2018, pp. 1–5.
- [6] D. Hallac, J. Leskovec, and S. Boyd, “Network lasso: Clustering and optimization in large graphs,” in *Proc. ACM SIGKDD*, Sydney, Australia, 2015, pp. 387–396.
- [7] A. H. Sayed, “Adaptation, learning, and optimization over networks,” *Found. Trends Mach. Learn.*, vol. 7, no. 4-5, pp. 311–801, 2014.
- [8] D. P. Bertsekas, “A new class of incremental gradient methods for least squares problems,” *SIAM J. Optim.*, vol. 7, no. 4, 1997.
- [9] A. Nedić, A. Olshevsky, A. Ozdaglar, and J. N. Tsitsiklis, “Distributed subgradient methods for multi-agent optimization,” *IEEE Trans. Automat. Contr.*, vol. 54, no. 1, pp. 48–61, 2009.
- [10] E. Rizk, R. Nassif, and A. H. Sayed, “Network classifiers with output smoothing,” in Available as arXiv:1911.04870, Oct. 2019.

- [11] R. Nassif, S. Vlaski, C. Richard, and A. H. Sayed, “Learning over multitask graphs—Part I: Stability analysis,” *IEEE Open Journal of Signal Processing*, vol. 1, pp. 28–45, 2020.
- [12] J. Chen, C. Richard, and A. H. Sayed, “Multitask diffusion adaptation over networks,” *IEEE Trans. Signal Process.*, vol. 62, no. 16, pp. 4129–4144, 2014.
- [13] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, “Proximal multitask learning over networks with sparsity-inducing coregularization,” *IEEE Trans. Signal Process.*, vol. 64, no. 23, pp. 6329–6344, 2016.
- [14] R. Nassif, C. Richard, A. Ferrari, and A. H. Sayed, “Multitask diffusion lms with sparsity-based regularization,” in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2015, pp. 3516–3520.
- [15] M. Belkin, I. Matveeva, and P. Niyogi, “Regularization and semi-supervised learning on large graphs,” in *Proc. Conf. Learning Theory*, Banff, Canada, 2004, pp. 624–638.
- [16] R. K. Ando and T. Zhang, “Learning on graph with laplacian regularization,” in *Proc. Adv. Neural Inf. Process. Syst.*, Cambridge, MA, USA, 2006, pp. 25–32.
- [17] X. Zhu and Z. Ghahramani, “Learning from labeled and unlabeled data with label propagation,” Technical report, 2002.
- [18] D. Zhou, O. Bousquet, T. Lal, J. Weston, and B. Schölkopf, “Learning with local and global consistency,” in *Proc. Adv. Neural Inf. Process. Syst.*, 2003, pp. 321–328.
- [19] H. Ambos, N. Tran, and A. Jung, “Classifying big data over networks via the logistic network lasso,” in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 855–858.
- [20] N. Tran, H. Ambos, and A. Jung, “Classifying partially labeled networked data via logistic network lasso,” in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2020, pp. 3832–3836.
- [21] A. Jung, “Clustering in partially labeled stochastic block models via total variation minimization,” in *2020 54th Asilomar Conference on Signals, Systems, and Computers*, 2020, pp. 731–735.
- [22] Y. SarcheshmehPour, Y. Tian, L. Zhang, and A. Jung, *Networked federated multi-task learning*, 2021.
- [23] Y. Sarcheshmehpour, M. Leinonen, and A. Jung, “Federated learning from big data over networks,” in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process*, Toronto, Canada, 2021, pp. 3055–3059.
- [24] D. W. Hosmer and S. Lemeshow, *Applied Logistic Regression*, 2nd. Wiley, NJ, 2000.
- [25] S. Theodoridis and K. Koutroubas, *Pattern Recognition*, 4th. Academic Press, 2008.

- [26] L. I. Rudin, S. Osher, and E. Fatemi, “Nonlinear total variation based noise removal algorithms,” *Physica D: Nonlinear Phenomena*, vol. 60, no. 1, pp. 259–268, 1992.
- [27] Y. Nesterov, *Introductory lectures on convex optimization: A basic course*. Springer, 2004.
- [28] D. Ciulin, “About sign function and some extensions,” in *Innovations and Advanced Techniques in Systems, Computing Sciences and Software Engineering*, E. Khaled, Ed., Springer, Dordrecht, 2008, pp. 148–153.
- [29] F. H. Clarke, *Optimization and Nonsmooth Analysis*. Wiley New York, 1983.
- [30] N. Perraudin, J. Paratte, D. Shuman, *et al.*, “GSPBOX: A toolbox for signal processing on graphs,” *ArXiv e-prints*, Aug. 2014.
- [31] E. Abbe, “Community detection and stochastic block models: Recent developments,” *Journal of Machine Learning Research*, vol. 18, no. 177, pp. 1–86, 2018.
- [32] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus, “Adaptive graph signal processing: Algorithms and optimal sampling strategies,” *IEEE Transactions on Signal Processing*, vol. 66, no. 13, pp. 3584–3598, 2018.
- [33] J. H. Lawrimore, M. J. Menne, B. E. Gleason, *et al.*, *Global Historical Climatology Network - Monthly (GHCN-M), Version 3*. NOAA National Centers for Environmental Information, 2011.
- [34] H. Lawrimore, M. J. Menne, B. E. Gleason, *et al.*, “An overview of the global historical climatology network monthly mean temperature data set, version 3, j,” 2011.
- [35] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs,” *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644–1656, 2013.
- [36] B. Xu, R. Huang, and M. Li, “Revise saturated activation functions,” *CoRR*, vol. abs/1602.05980, 2016.
- [37] M. M. Lau and K. Hann Lim, “Review of adaptive activation function in deep neural network,” in *2018 IEEE-EMBS Conference on Biomedical Engineering and Sciences (IECBES)*, 2018, pp. 686–690.
- [38] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feed-forward neural networks,” in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Y. W. Teh and M. Titterton, Eds., ser. Proceedings of Machine Learning Research, vol. 9, Chia Laguna Resort, Sardinia, Italy: PMLR, May 2010, pp. 249–256.