Online Federated Learning over decentralized networks

Zhang, Chi

2018

Zhang, C. (2018). Online Federated Learning over decentralized networks. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/106445>

<https://doi.org/10.32657/10220/47928>

**NANYANG
TECHNOLOGICAL
UNIVERSITY**

**ONLINE FEDERATED OPTIMIZATION
OVER DECENTRALIZED NETWORKS**

CHI ZHANG

INTERDISCIPLINARY GRADUATE SCHOOL

2018

ONLINE FEDERATED OPTIMIZATION OVER DECENTRALIZED NETWORKS

CHI ZHANG

INTERDISCIPLINARY GRADUATE SCHOOL

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirement for the degree of
Doctor of Philosophy

2018

Statement of Originality

I hereby certify that the intellectual content of this thesis is the product of my original research work and has not been submitted for a higher degree to any other University or Institution.

Mar. 2019

.....

Date

Chi Zhang

.....

Chi ZHANG

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

Mar. 2019

.....

Date

.....

Prof Yeng Chai SOH

Authorship Attribution Statement

This thesis contains material from papers published in the following peer-reviewed journals.

Chapter 5 is published as ,Zhang, Chi and Zhao, Peilin and Hao, Shuji and Soh, Yeng Chai and Lee, Bu Sung and Miao, Chunyan and Hoi, Steven CH. Distributed multi-task classification: a decentralized online learning approach, Machine Learning Journal (2017).

The contributions of the co-authors are as follows:

- Dr Peilin Zhao provided the initial project direction and edited the manuscript drafts.
- I co-designed the study with Dr Hao Shuji and performed all the laboratory work at the lab of Living for the Elderly. I also analyzed the data.
- Prof Soh Yeng Chai, Prof Lee Bu Sung, Prof Miao Chunyan and Prof Steven Hoi revised my manuscript drafts..
- All Simulations, Data Validations including sample preparation, were conducted by me.

Mar. 2019

.....

Date

.....

Chi ZHANG

Acknowledgements

Almost all the thesis starts with an acknowledgement and I finally know why this is so. Without these people, this work could never be done, and these four years could be totally different. All the way through the past four years, I am so grateful to meet the right people at the right time.

First of all, I would like to thank my supervisor Prof Soh Yeng Chai and my co-supervisor Prof Lee Bu Sung for their consistent help in the past four years. Prof Soh and Prof Lee led me to the world of scientific research and gave me the freedom to explore various directions. Their enthusiasm and patience guide me through confusion and frustration, and their sharp insights for various problems always motivate me to move forward. Being able to be supervised by them is and will always be my greatest honor, and all these words can't express my deep thanks to them.

I would like to show my sincere appreciation to my research supervisors and leaders Dr. Zhao Peilin and Prof Steven Hoi. When first met them, I was only a freshman with only limited knowledge of machine learning, and they opened the door of online learning for me and consistently provided talented ideas. Their ways of doing research have deeply influenced my research philosophy. What can be better than meeting and working with such talented leaders?

As a Ph.D. student, I am very grateful to my school: Interdisciplinary Graduate School where I have joined interesting courses, attended various seminars in multiple domains. As an IGS student club member, I was also very lucky to organize various events with my friends: Shao Hongxin, Nguyen Duy Tam, Betty Fan and Huan Tan.

Finally, I would like to thank my parents for their ever-lasting love and support. It is their encouragements and strong belief that always support me to pursue my dreams in Singapore.

Abstract

Online Federated Learning (OFL) refers to the online optimization that is distributed over a decentralized network while still seek for training high-quality models. It differs from the training methods in centralized distributed optimization where information from different nodes needs to be sent to a central unit (*e.g.*, the server-and-worker framework), OFL allows each node to perform local operation (*e.g.*, train local model) and only contacts with its immediate neighbors, liberating it from the control of the ‘master’ node. Therefore, the computation and communication are totally decentralized, avoiding the traffic congestion and network coordination problems that are inevitable to most centralized distributed optimization. Such a learning paradigm finds its applications in a variety of research areas, such as sensor network estimation, multi-agent coordination, distributed tracking and source scheduling.

Although being extensively researched, we address several critical problems and their corresponding solutions to make OFL more practical and more efficient in real-scenarios in this thesis. Performance of OFL algorithms is known to be affected by the mixing ability of the underlying graph, and well-connected networks (*e.g.*, complete graphs) are naturally preferred for machine learning problems since it guarantees sufficient information exchange on each round and leads to faster convergence rates. However, on the other side of the coin is the heavy computation and communication cost when combining neighborhood parameters, which clearly dominate the local computation cost under online settings. To overcome this limitation, we propose an edge sampling algorithm to replace the costly deterministic communication cost with a stochastic strategy. Compared with existing solutions, our sampling method enjoys the merits of unbiasedness, low variance and graph-adaptation. These properties allow the new algorithm to escape from the heavy communication burden and guarantee it to be more

communication-efficient, which are proved with theoretical convergence rates and validated with synthetic and real-world datasets in this thesis.

In the meanwhile, most of the existing OFL algorithms often make the assumption that there are no local or global constraints, which clearly limit their practical use in real scenarios. Local constraints are quite common in optimization problems, such as budget limitation in portfolio selection, prior information adaptation in data fitting and projection into feasible areas in machine learning. To get an accurate solution in this case, one practical solution would be constructing a local Lagrangian formulation and solving the convex-concave problem by iterative updating local variables and communicating with neighbors. In this thesis, we answer this challenge by developing and analyzing an algorithm to seek for the optimal saddle point for decentralized online convex-concave problems. With clear upper and lower regret bounds, we theoretically guarantee the performance of our proposed algorithm, as well as revealing its relationship with the network size and topology. We validate our analysis by performing experiments for online AUC maximization, which turns out to be a byproduct of our general solution but stands out as the first distributed online AUC maximization algorithm.

Finally, we consider the case that is often neglected by previous OFL algorithms, where the optimal values for different machine learning models may only be similar rather than identical. We present a new learning framework and jump out of the stereotype of limiting each learning node to a ‘worker’ for a specific model. A more general setting is therefore proposed where each learning unit is able to learn all models, and knowledge is transferred in two phases: model information is shared within each node in the first phase, and then all models collaborate together to achieve a common minimizer. The proposed algorithms successfully avoid suffering the challenge of ‘Pareto optimality’ since the potential optimum for all nodes is identical rather than similar.

Contents

Acknowledgements	vii
Abstract	ix
Contents	xi
List of Figures	xiv
List of Tables	xv
1 Introduction	1
1.1 Motivations	1
1.2 Research Scope and Contributions	4
1.2.1 Online Federated Learning with Edge Sampling	4
1.2.2 Online Federated Convex-Concave Learning	5
1.2.3 Online Federated Multi-task Learning	6
1.3 Thesis Organization	7
1.4 Notation Table	7
2 Related Work	8
2.1 Online Learning	8
2.1.1 General Framework	9
2.1.2 Updating Protocols	10
2.1.2.1 First-order OL Algorithm	10
2.1.2.2 Second-Order OL Algorithm	12
2.2 Decentralized Optimization	13
2.2.1 Primal Solutions	14
2.2.2 Dual Solutions	16
2.3 Multi-task Optimization	17
3 Online Federated Learning with Edge Sampling	20
3.1 Existing Solutions	21
3.2 Online Federated Learning with Edge Sampling	23
3.2.1 Sampling Strategy	23
3.2.2 Comparing Node Sampling with Edge Sampling	25
3.2.3 Implementations of Edge Sampling	25

3.3	Convergence Analysis	26
3.3.1	Convergence Analysis	26
3.3.2	Communication Comparison	26
3.3.3	Discussion on Graphs	28
3.3.3.1	N-complete graph	28
3.3.3.2	D-regular Graph	29
3.3.3.3	Star Graph	30
3.3.3.4	General Case	31
3.4	Experiments	31
3.4.1	Experimental Setup	31
3.4.2	Mixing Rate of Markov Chain	32
3.4.3	Binary Classification	33
3.5	Budget-Efficient Decentralized Optimization	35
3.6	Chapter Summary	35
3.7	Appendix: Proof for Theorem 3.1	36
4	Online Federated Convex-Concave Optimization	42
4.1	Decentralized Online Saddle Point Algorithm	43
4.1.1	Preliminaries	43
4.1.2	Problem Formulation	44
4.1.2.1	Auxiliary Sequences	46
4.2	Theoretical Analysis	47
4.2.1	Network Error Bound	47
4.2.2	Upper Bound	48
4.2.3	Lower Bound	51
4.3	Experiments	52
4.3.1	Distributed Online AUC Maximization	52
4.3.2	Algorithms	53
4.3.3	Datasets and Networks	53
4.3.4	Experimental Results	54
4.4	Chapter Summary	57
4.5	Appendix: Proof for Lemmas	58
4.5.1	Jensen's Inequalities	58
4.5.2	Proof for Lemma 4.1	58
4.5.3	Proof for Lemma 4.2	59
4.5.4	Proof for Lemma 4.5	62
4.5.5	Proof for Eq (4.15)	64
5	Online Federated Learning for Decentralized Multi-task Optimization	66
5.1	Motivations and Existing Solutions	67
5.2	Online Federated Learning for Decentralized Multi-task Optimization (DOM)	70
5.2.1	Problem Formulation	70
5.2.2	Node Update	71

5.2.2.1	Node updating rule	72
5.2.2.2	Selection of relationship matrix B	73
5.2.3	Nodes Communication	75
5.3	Theoretical Analysis	77
5.3.1	Proof Setup	77
5.3.2	Weight Difference and Regret Bound	78
5.4	Experiment	81
5.4.1	Benchmarks	81
5.4.2	Performance Analysis	82
5.5	Chapter Summary	85
5.6	Appendix: Proof for Lemmas	86
5.6.1	Proof for Eq (5.18)	86
5.6.2	Proof for Lemma 5.4	86
5.6.3	Proof for Lemma 5.5	87
5.6.4	Proof for Lemma 5.6	88
5.6.5	Proof for Theorem 5.7	90
6	Conclusion and Future Work	92
6.1	Conclusion	92
6.2	Future Work	93
6.2.1	Sufficient Updating Local Model	93
6.2.2	Non-convex Optimization	94
A	List of Publications	96
	Bibliography	97

List of Figures

1.1	The ‘master-worker’ framework in centralized distributed optimization and ‘federated-learning’ in decentralized distributed optimization. . . .	2
3.1	Mixing rate of Markov chain on different graphs with the gap referring to $\ W^t - \mathbf{1}_n/n\ _2$. Left is the result on the k-connected cycle with $n = 1000$, $k = 75$ and $z = 0.25$; right is the result on the random graph with $n = 1000$, $c = 0.15$ and $z = 0.05$	33
3.2	Distributed binary classification on different graphs with the gap referring to $f(\hat{\mathbf{w}}_1^{T'}) - f(\mathbf{w}^*)$. Left is the result on the k-connected cycle with $n = 1000$, $k = 75$ and $z = 0.25$; right is the result on the random graph with $n = 1000$, $c = 0.15$ and $z = 0.05$	34
4.1	Effect of node numbers for online learning process on <i>a9a</i> and <i>w8a</i> datasets.	56
4.2	Effect of graph types for online learning process on <i>a9a</i> and <i>w8a</i> datasets.	57
5.1	Two tasks with different optimal value $\mathbf{w}_1^*$ and $\mathbf{w}_2^*$. The network will finally converge to some point $\mathbf{w}$ between $\mathbf{w}_1^*$ and $\mathbf{w}_2^*$. Obviously, any attempt to decrease one task’s loss lies at the cost of increasing the other’s (e.g., moving $\mathbf{w}$ down along the red line increases $J_2(\mathbf{w})$). . . .	68
5.2	Above is the learning process on Sentiment dataset; below is the parameter effect on Synthetic and MHC-I dataset.	84

List of Tables

1.1	Notation in the Thesis	7
3.1	Algorithm Comparison	22
4.1	Datasets Information	54
4.2	Experimental results(%) with different network types and scales	55
5.1	Error Prediction Rate(%) and Standard Deviations(%) on Real-world Datasets	83

Chapter 1

Introduction

1.1 Motivations

In the past few decades, the ‘Big Data’ era has brought new opportunities and challenges, leading to an upsurge of academic research and industrial applications. It holds great promises for discovering subtle population patterns and heterogeneities that are not possible with small-scale data, while on the other hand its massive sample size and high dimensionality lead to computational and statistical challenges for training machine learning models. Machine learning algorithms for big data often suffer from extremely long computation time and large storage space when dealing with big data. Datasets often become too large to be processed by any single unit and it’s favorable to distribute smaller subsets to multiple processors, so that the computation overload is locally light. A canonical example is the problem of Kernel regression on a large dataset [1], where the kernel matrix often becomes too large to fit in a single unit’s capacity and it’s better to distributed datasets to multiple processing nodes.

These opportunities and challenges have motivated research of distributed learning over multiple nodes. Based on the adopted strategy, the distributed machine learning algorithms can be roughly divided into two categories: centralized and decentralized optimization.

Figure 1.1: The ‘master-worker’ framework in centralized distributed optimization and ‘federated-learning’ in decentralized distributed optimization.

Parallel applications to solve optimization problems on a centralized network has been intensively studied in [2–4]. These applications generally use a ‘master-worker’ paradigm where a single master node first dispatches a subset of computation to multiple worker nodes and then gathers computed results from the worker process. Having such a centralized control also eases the burden of adapting to a heterogeneous environment since only the master node needs to be concerned with the matchmaking process of assigning tasks to make the best use of the resource characteristics [4].

However, despite its broad success in handling large datasets, researchers and engineers soon notice that the performance of this centralized distributed optimization can be affected by many factors. First of all, the communication overhead between the master and worker often leads to traffic jam and degrades performance. Also, the processing procedure on the master node is often the bottleneck for the whole graph since it needs to process information from all workers. In many applications, datasets may naturally be observed or distributed geographically and sending all data to a fusion center leads to extra transportation cost, or even be impractical due to political, privacy-sensitive or technological reasons. A typical example arises in the development of medical predictive models with the collaboration of multiple agents (*e.g.*, hospitals, medical databases), where model performance is often proportional to the number of instances used, while external data utilization or validation are prohibited. In this case, instead of gathering data to a fusion center, a common approach to preserve data privacy is establishing local personalized models and then collaborating over decentralized networks by sending models to the neighborhood rather than the data itself [5].

These challenges have motivated the studies of optimization over decentralized networks, where the global center is removed and each node only contacts its immediate neighbors so the computation and communication are totally decentralized, avoiding the traffic jam and network coordination problems mentioned above. Such an optimization method is often called ‘federated learning’ [6] since each node is now liberated from the control of the master node and all nodes collaborate to reach consensus by sharing information with their direct neighbors without the central coordination. This federated learning pursues the goal of pure *localized* operation on a predefined network, such as internal computation and neighborhood communication. It finds its application for both machine learning research and applications, including sensor network estimation [7], multi-agent coordination [8, 9], distributed tracking [10] and source scheduling [11]. It may also cover applications from other domains, since nowadays the *network* may also refer to social networks, power networks, transportation networks and biological networks [12].

Apparently, any (online) federated learning algorithm should answer the following questions when optimizing over decentralized networks: (1) how do each node fit into local data so the model is locally efficient; (2) how to collaborate with neighbors to reach global consensus and avoid local overfitting. The common approach is decomposing the learning into two phases, namely ‘combination phase’ and ‘adaptation phase’, by iteratively updating local information and exchanging neighborhood knowledge. Taking the commonly-used ‘Distributed Gradient Descent’ (DGD) algorithm as an example, node i first combines parameters with a weighted average of the parameters from its neighborhood:

$$\text{Combine : } \mathbf{v}_i^t = \sum_{j \in N(i)}^n a_{i,j} \mathbf{w}_j^t. \quad (1.1)$$

The next local iterate $\mathbf{w}_i^{t+1}$ is chosen by minimizing an averaged first-order approximation to the function f_i with a projection step to ensure the primal parameters still lie in the feasible region:

$$\text{Adapt : } \mathbf{w}_i^{t+1} = P_{\Omega} [\mathbf{v}_i^t - \eta^t \cdot \partial f_i(\mathbf{v}_i^t)]. \quad (1.2)$$

In this thesis, we shall focus on the *online settings* where data arrives in a sequence order and each node receives one instance $\mathbf{x}_j^t$ and updates its $\partial f_i(\mathbf{v}_i^t)$ based on the online gradient $\partial f_i(\mathbf{v}_i^t) = \frac{\partial f_i(\mathbf{v}_i^t, \mathbf{x}_j^t)}{\partial \mathbf{v}_i^t}$. This leads to the ‘Distributed Online Gradient Descent’ (D-OGD) learning, which can be regarded as a stochastic version of the general DGD framework. Such an online learning structure enjoys the merits of low computation cost per round and low memory requirement for each node, and therefore suits for many real scenarios. Taking the ‘Wireless Sensor Networks’ (WSN) as an example, each node may only be a simple detecting sensor with limited computation ability and storage space. Online learning allows the sensor node on WSN to quickly response the incoming instance and reaches a more precise measurement by collaboration (*e.g.*, averaging the results). In the meanwhile, in many scenarios, machine learning models built on a fixed dataset cannot keep pace with the fluctuation of data and their generalization performance degrades as time goes by, and online learning does not have to assume the data to be stationary and adapts itself to the pattern of the streaming data.

Finally, it’s worth mentioning that many of our results can also be applied to the offline settings (*e.g.*, sampling technique in Chapter 3) although our studies mainly focus on the online setting in this thesis.

1.2 Research Scope and Contributions

Our research focuses on developing new techniques and solutions to improve current results on online decentralized networks, spanning over a number of domains as follows.

1.2.1 Online Federated Learning with Edge Sampling

As mentioned in the previous section, online decentralized optimization generally uses the updating protocols by iteratively using internal computation and external communication to train high-quality models. Performance of these models is known to be affected by the mixing ability of the underlying graph. Well-connected networks (*e.g.*, complete graphs) are naturally preferred than poorly-connected networks (*e.g.*, cycle

graphs) for machine learning problems since it guarantees sufficient information exchange on each round and leads to faster convergence rates. However, on the other side of the coin is the heavy computation and communication cost when combining neighborhood parameters, which clearly dominate the local computation cost under online settings. Taking learning on fully-connected graphs as an example, its $\mathcal{O}(nd)$ the computation cost of combining weights from its direct neighbors clearly exceeds the following $\mathcal{O}(d)$ cost for the local computation when the projection can be sufficiently computed.

To reduce the communication cost in the adaptation step, we propose a decentralized optimization algorithm by replacing the costly deterministic communication with a stochastic strategy based on an edge sampling method. On each round, each node samples a subset of its neighbors instead of communicating with all the neighbors. The sampling rate is designed to be optimal and graph-dependent so that it can avoid redundant communication for well-connected networks while still keep most connections in poorly-connected networks.

Compared with existing solutions, our sampling method enjoys the merits of unbiasedness, low variance and graph-adaptation. These properties allow the new algorithm to escape from the heavy communication burden and guarantee it to be more communication-efficient, which are proved with theoretical convergence rates and validated with synthetic and real-world datasets in this thesis. Taking the d -regular graph as an example, our algorithm is shown to be $\mathcal{O}(d)$ times faster than the unsampled algorithm when giving the same communication budget, which is also confirmed by the following experiments.

1.2.2 Online Federated Convex-Concave Learning

In the previous research, local or global constraints are often neglected so that the local updating resembles the online convex learning (OCL) for simplicity. However, local constraints are quite common in optimization problems, such as budget limitation in portfolio selection, prior information adaptation in data fitting and projection into feasible areas in machine learning. One practical solution for such a constrained problem

would be constructing a local Lagrangian formulation and solving the convex-concave problem by iterative updating, and then sharing local weights via communication.

In this thesis, we develop and analyze an algorithm to seek for the optimal saddle point for online decentralized convex-concave problems. Our theoretical analysis guarantees the convergence of the proposed algorithm with clear upper and lower regret bounds, as well as revealing its reliance on the network size and topology. We validate our analysis by performing experiments for online AUC maximization, which turns out to be a byproduct of our general solution but stands out as the first distributed online AUC maximization algorithm.

1.2.3 Online Federated Multi-task Learning

Most of the current decentralized studies focus on designing algorithms to seek for an identical solution for all nodes on the graphs by adopting various communication protocols, which guarantees the general convergence and avoids local overfitting. Such a learning target fails to be applied in the cases when local models may only be similar rather than identical. For instance, although a general trend may exist in designing recommendation systems or analyzing trends for stock markets, the learning model should also consider each user's preferences and each stock's individual characters.

We present a new learning framework and jump out of the stereotype of limiting each learning node to a 'worker' for a specific model. A more general setting is therefore proposed where each learning unit is able to learn all models, and knowledge is transferred in two phases: model information is shared within each node in the first phase, and then all models collaborate together to achieve a common minimizer. The proposed algorithms successfully avoid suffering the challenge of 'Pareto optimality' since the potential optimum for all nodes is identical rather than similar.

1.3 Thesis Organization

The remainder of this thesis is organized as follows. In Chapter 2, we review some existing works that are related to our online federated learning. In Chapter 3, we study the general convex optimization problems and propose a computation-communication imbalanced issue for online learning. We provide a solution based on a novel sampling strategy named as edge sampling. In Chapter 4, we propose an extension of the generally used ‘combine-and-then-adapt’ strategy to the convex-concave cases, and show the relationship between online regret and network topology and size. In Chapter 5, we further consider the scenario of reaching consensus for online multi-task learning and develop a new framework by decomposing the communication into two phases: internal communication for multi-task information sharing and external communication for reaching consensus. Finally, we conclude the thesis in Chapter 6 and discuss several future directions.

1.4 Notation Table

In order to make the presentation more clear, we show some common notations in the following table.

Notation	Explanation
$\mathbf{x}$	a single point, typically belonging to $\mathbb{R}^d$
$\mathcal{X}$	domain from which the samples come from
$\mathbf{w}$	a weight vector, typically belonging to $\mathbb{R}^d$
Ω	domain for weight $\mathbf{w}$ belongs to
y	the label of $\mathbf{x}$
$\ell(\cdot)$	the loss function defined on $\mathbb{X}$
G	a graph, with vertex set $V = \{1, \dots, n\}$ and edge set $E \subseteq V \times V$
A	communication matrix for graphs, with its component $a_{i,j}$ represents the communication weight from node i to node j
$\lambda(A)$	eigenvalues of matrix A , in a decreasing order
p	sampling rate
$\mathbb{E}$	expectation of variables or matrices

TABLE 1.1: Notation in the Thesis

Chapter 2

Related Work

Our online federated learning is closely related to a few research areas including online learning, federated learning on decentralized networks and multi-task learning. In this chapter, we will separately review existing studies that are relevant to this thesis.

2.1 Online Learning

Online learning (OL) is a well-established learning paradigm with both theoretical and practical appeals [13]. The goal of online learning is to make a sequence of accurate predictions given knowledge of the correct answer to previous prediction tasks. The learning purely relies on the current instance without the necessities of storing previous instances, as opposed to the batch learning techniques which generate the best predictor by learning on the entire training datasets. Therefore, it enjoys the merits of low computation cost per round and low storage requirement.

In this part, we provide an overview of the general framework of online learning, followed by both first and second order updating protocols in recent studies.

2.1.1 General Framework

Taking online binary classification algorithm as an example, it's performed in a sequence of consecutive rounds, where on each round the classifier receives an instance $\mathbf{x}_t \subseteq \mathbb{R}^n$ and is required to apply its current prediction rule to generate a label $\hat{y}_t$ from a target domain $\mathcal{Y}$. After prediction the answer, the true label y_t is revealed to the classifier and the learner suffers from a loss $\ell_t(y_t, \hat{y}_t)$, which is used to measure the discrepancy between its final prediction and the correct one. Upon receiving the loss, the classifier renews its prediction rule by updating its weight matrix and then proceeds to the next round. Algorithms 1 summaries the whole updating process.

Algorithm 1 General Framework for Online Learning

```

Initialize :  $\mathbf{w}_0 = 0$ 
for  $t = 1, \dots, T$  do
  The classifier receives an instance:  $\mathbf{x}_t \subseteq \mathbb{R}^n$ 
  The classifier makes a prediction:  $\hat{y}_t = f(\mathbf{x}_t; \mathbf{w}_t)$ 
  The true label is revealed  $y_t \in \mathcal{Y}$ 
  The classifier suffers from loss:  $\ell_t(y_t, \hat{y}_t)$ 
  if  $l(\mathbf{w}_t; (\mathbf{x}_t, y_t)) > 0$  then
    The classifier updates its classification model:
     $\mathbf{w}_{t+1} \leftarrow \mathbf{w}_t + \Delta(\mathbf{w}_t; (\mathbf{x}_t, y_t))$ 
  end if
end for

```

Overall, the goal of online learning is to minimize the cumulative error over the entire sequence of data compared with the best model in hindsight. This is often captured by the *regret* of the algorithm, which measures the gap between trained classifier and the optimal one and is also known as how “sorry” the classifier is not able to follow the optimal prediction $\mathbf{w}^*$:

$$\text{Regret}_T = \sum_{t=1}^T \ell_t(y_t, \hat{y}_t) - \sum_{t=1}^T \ell_t(y_t, f(\mathbf{x}_t; \mathbf{w}^*)). \quad (2.1)$$

The optimal prediction $\mathbf{w}^*$, although unknown in most scenarios, is often assumed to be a fixed value in the feasible region Ω with respect to the minimal cumulative loss

to the entire learning process:

$$\mathbf{w}^* = \arg \min_{\mathbf{w} \in \Omega} \sum_{t=1}^T \ell_t(y_t, f(\mathbf{x}_t; \mathbf{w})). \quad (2.2)$$

If the regret grows sub-linearly with the number of rounds T (e.g., $\mathcal{O}(\sqrt{T})$ or $\mathcal{O}(\log(T))$), which implies the average loss tends to zero as T is sufficiently large, the online learning classifier is guaranteed to reach the optimal value $\mathbf{w}^*$.

2.1.2 Updating Protocols

The core part of online learning algorithms is how the classifier updates its stored weight $\Delta(\mathbf{w}_t; (\mathbf{x}_t, y_t))$. Based on the updating protocols, most of the algorithms can be roughly divided into two categories, namely first-order algorithms and second-order algorithms.

2.1.2.1 First-order OL Algorithm

First-order optimization refers to the setting where the classifiers only use first order (sub)gradient information to optimize its objective function.

Perceptron: The Perceptron algorithm [14, 15] is the oldest algorithm for online learning, with its weight updated as $\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \mathbf{x}_t$ if an error occurred. In the later research, many variants of Perceptron have been proposed to further improve its performance. Among them, the simplest modification is "Normalized Perceptron" algorithm by updating in a normalized way:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + y_t \mathbf{x}_t / \|\mathbf{x}_t\|.$$

Mistake bound the algorithm is improved from $\left(\frac{R}{\gamma}\right)^2$ to $\left(\frac{1}{\gamma}\right)^2$ for the separable case due to the normalization effect.

PA: The fixed updating rule is then modified by the Online Passive-Aggressive (PA) [16] by using an adaptive updating strategy:

$$\mathbf{w}_{t+1} = \arg \min_{\mathbf{w} \in \Omega} \frac{1}{2} \|\mathbf{w} - \mathbf{w}_t\|_2^2 \quad s.t. \quad \ell_t(\mathbf{w}) = 0.$$

This optimization problem admits a closed-form solution for hinge-loss:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \tau y_t \mathbf{x}_t, \quad \tau_t = \ell_t / \|\mathbf{x}_t\|_2^2.$$

Based on the updating strategy, PA contains a major difference from Perceptron algorithm: Perceptron algorithms only make an update when a classification mistake occurs, while PA algorithms make an update whenever the loss is nonzero (even when the classification is correct), leading to the better performance in empirical results.

OGD: Online Gradient Descent (OGD) [17], which can be viewed as an online version of Stochastic Gradient Descent (SGD), is another popular online learning algorithm for its simplicity and high-efficiency. Different from the PA with a local optimal learning rate τ_t at each round, OGD employs some predefined learning sequences $\{\eta_t\}$ and decompose the learning into two phases:

(1) the algorithm first updates its stored weight in the direction of the gradient of the current loss:

$$\mathbf{w}_{t+\frac{1}{2}} = \mathbf{w}_t - \eta_t \nabla \ell_t(\mathbf{w}_t);$$

(2) the updated weight is projected into the feasible region:

$$\mathbf{w}_{t+1} = \arg \min_{\mathbf{w} \in \Omega} \|\mathbf{w} - \mathbf{w}_{t+\frac{1}{2}}\|.$$

Due to the similarity to updating in batch learning, OGD often enjoys simpler regret analysis, and results in [17, 18] show that OGD algorithm achieves sublinear regret $\mathcal{O}(\sqrt{T})$ for the general convex case and logarithm regret $\mathcal{O}(\log T)$ in the strongly convex case, with respect to the best optimizer in hindsight.

Others: In literature, there are also some other first-order online learning algorithms including algorithms to seek for the maximum margin including Approximate

Large Margin Algorithms (ALMA) [19] and Relaxed Online Maximum Margin Algorithm (ROMMA) [20] or consider multiplication algorithms including Online Weighted Majority (OWM) [21] and Follow the Regularized Leader (FTRL) [22].

2.1.2.2 Second-Order OL Algorithm

Unlike the first-order online learning algorithms that only employ the first order derivatives information, second-order online learning algorithms also exploit extra information including data (or weight) distribution or Hessian matrix of optimization objectives to accelerate convergence rates.

SOP: Cesa and Gentile [23] first points out that the greedy update in Perceptron may significantly change the stored parameter $\mathbf{w}_t$ when the coming instance $\mathbf{x}_t$ is close to the distribution boundary, and they present a second-order Perceptron (SOP) algorithm by storing data-correlation matrix and preprocess the data before feeding into the classifier. To be specific, the algorithm stores the correlation matrix $M = \sum_{t=1}^T \mathbf{x}_t \mathbf{x}_t^\top$ and normalize the learning sequence to $(M^{-1/2} \mathbf{x}_t, y_t)$. By reducing the correlation matrix of the coming instances, they show SOP algorithm can achieve significantly better mistake bounds.

CW: Dredze [24] presents a second-order online learning named as Confidence-Weighted learning (CW) that maintains a probabilistic measure of confidence in parameter $\mathbf{w}_t$ rather than scalars. The algorithm draws a parameter vector $\mathbf{w} \sim N(\mu, \Sigma)$ with its component μ_j representing the ordinary weight and σ_j represents the confidence. The smaller σ_j is, the more confidence we have in the mean μ_j . Similar to PA, the updating ensures its weight distribution is close to the previous while the prediction is sufficiently correct:

$$(\mu_{t+1}, \Sigma_{t+1}) = \arg \min_{\mu, \sigma} D_{KL}(\mathcal{N}(\mu, \sigma) \mid \mathcal{N}(\mu_t, \sigma_t)),$$

$$s.t. \quad \Pr[y_t \mathbf{x}_t \mathbf{w}_t \geq 0] \geq \eta,$$

where D_{KL} refers to KL-divergence of distribution and η is a fixed parameter.

A few variants of CW are studied in the later research either adding new adaptive regularization (AROW in [25]) or exploit different margins for different instances (SCW in [26]). The goal of these research is to make CW more adaptive to the data and robust to noisy by preventing significant changes to the current value.

SON: In practice, second-order online learning algorithms often converge faster and therefore performs better in real scenarios, while on the other hand, its storage and computation requirement is much larger than its first-order counterparts. This problem is partially alleviated in Sketched Second Order Newton (SON) [27], where the authors adopt various sketching techniques to make the computation linear in the sparsity of features.

As a conclusion, we briefly review the recent progress in online learning, and the reader is referred to the complete survey of online learning in [28] with their actual performance in [29].

2.2 Decentralized Optimization

Decentralized optimization [30, 31] focuses on the development and analysis of solving optimization problems that are defined over a network $G = (V, E)$ with vertex set $V = \{1, \dots, n\}$ and edge set $E \subseteq V \times V$. Each node $i \in V$ stores its own parameters $\mathbf{w}_i \in \mathbb{R}^d$ and uses the associated local function $f_i : \mathbb{R}^d \rightarrow \mathbb{R}$ to evaluate the performance of its parameters. The communication between nodes is specified by the graph G : node i can only directly communicate with its immediate neighbors $N(i) = \{j \in V \mid (i, j) \in E\}$ through a combination matrix A .

The overarching goal of decentralized optimization is minimizing a global objective defined by the average over the local functions:

$$f(\mathbf{w}) = \frac{1}{n} \sum_{i=1}^n f_i(\mathbf{w}) \quad s.t. \quad \mathbf{w} \in \Omega. \quad (2.3)$$

Unlike centralized distributed optimization where information from different computation nodes need to be sent to a central unit (e.g., the server-and-worker framework in [2]), the communication is purely decentralized and each node only contacts its immediate neighbors, which clearly avoids the traffic jam and coordination problems [12]. It finds its applications when datasets are too large to be processed by any single processor and it's favorable to distribute smaller subsets to multiple processing units so that the computation overload is locally light. Besides, it is often infeasible for any single agent to access the entire database due to political, economic, privacy-sensitive or technological reasons. For instance, problems such as sensor network estimation [7], multi-agent coordination [9] and distributed tracking [10] often assume that estimation or observation units are spread over the network and all nodes collaborate by sharing their local information to ensure better performance and increased fault tolerance, as well as local privacy protection. In these scenarios, datasets are distributed geographically that sending all data to a central unit often leads to extra transportation and communication cost [32].

Due to the necessities of designing distributed algorithms to store and process data on individual nodes, a series of distributed convex optimization algorithms have been proposed since the seminal work in [30]. Based on the techniques used, most of the *decentralized optimization* algorithms can be roughly divided into two categories: (1) primal solutions; (2) dual solutions.

2.2.1 Primal Solutions

Decentralized optimization with primal variables refers to the paradigm of directly exchanging individual weights with neighbors. Each node on the networks maintains its own parameters and is required to combine neighbor's parameters before (or after) local adaptation.

DGD: As we have reviewed in the introduction part, the typical "Decentralized Gradient Descent" (DGD) [12] algorithm decompose the learning process into two phases:

$$\begin{aligned} \text{Combine : } \quad \mathbf{v}_i^t &= \sum_{j \in N(i)}^n a_{i,j} \mathbf{w}_j^t. \\ \text{Adapt : } \quad \mathbf{w}_i^{t+1} &= P_{\Omega} [\mathbf{v}_i^t - \eta^t \cdot \partial f_i(\mathbf{v}_i^t)]. \end{aligned}$$

The later research [33] reveals the fact that its convergence rate depends on the stepsize. For a constant stepsize $\eta^t = \alpha$, DGD guarantees the network-wide mean solution to reduce at rates of $\mathcal{O}(1/t)$ until it reaches a level of $\mathcal{O}(\alpha)$ for the general convex cases. The final solution will fluctuate around the optimal solution due to the variance of the variance of each node. On the other hand, when adopting a decreasing stepsize $\eta^t = \mathcal{O}(1/\sqrt{t})$, DGD ensures the network-mean to strictly reach its global optimum $\mathbf{w}^*$, but with a slower rate $\mathcal{O}(1/\sqrt{t})$. Namely, there is a trade-off between convergence rate and convergence precision for DGD algorithms.

EXTRA: This problem is solved by the following research in [34], where the authors propose an exact first-order algorithm named as EXTRA by using two-step optimization. The updating protocol is modified as:

$$\mathbf{w}^{t+2} = (I + A)\mathbf{w}^{t+1} - \tilde{A}\mathbf{w}^t - \alpha(\nabla f(\mathbf{w}^{t+1}) - \nabla f(\mathbf{w}^t)),$$

with A representing the combination matrix with $a_{i,j}$ denoting its component and $\tilde{A} = (A + I)/2$.

EXTRA adopts a constant stepsize α and obtains an ergodic convergence rate of $\mathcal{O}(1/t)$ in the general convex case, while its variance reduction techniques guarantee it to reach the exact optimal solution. To see the later property, we can simply replace $\mathbf{w}^{t+2}, \mathbf{w}^{t+1}, \mathbf{w}^t$ with the optimal solution $\mathbf{w}^*$ and notice the equation still holds.

In literature, a few extensions of EXTRA are also proposed, either to apply this technique to the proximal case [35] or use a double variance reduction techniques to obtain faster convergence rate [36].

2.2.2 Dual Solutions

Decentralized optimization with dual variables (or primal-dual variables) refers to the setting of storing and exchanging dual parameters instead of the weights.

DDA: The typical "Decentralized Dual Averaging" (DDA) algorithm [37] follows the two-phase learning:

$$\begin{aligned} \text{Combine: } \mathbf{z}_i^{t+1} &= \sum_{j \in N(i)} a_{i,j} \mathbf{z}_j^t + \partial f_i(\mathbf{w}_i^t), \\ \text{Adapt: } \mathbf{w}_i^{t+1} &= \Pi_{\Omega}^{\psi}(\mathbf{z}_i^{t+1}, \eta^t) = \arg \min_{\mathbf{w} \in \Omega} \left\{ \langle \mathbf{z}_i^{t+1}, \mathbf{w} \rangle + \frac{1}{\eta^t} \psi(\mathbf{w}) \right\}. \end{aligned}$$

In particular, node i first combines the new dual parameters $\mathbf{z}_i^{t+1}$ from a weighted average of its own subgradient $\partial f_i(\mathbf{w}_i^t)$ and the parameters from its neighborhood. The next local iterate $\mathbf{w}_i^{t+1}$ is chosen by minimizing an averaged first-order approximation to the function f_i with a proximal function ψ to ensure the primal parameters do not oscillate wildly. Typical examples of ψ include ℓ_1 regularization $\psi(\mathbf{w}) = \|\mathbf{w}\|_1$, ℓ_2 regularizations $\psi(\mathbf{w}) = \|\mathbf{w}\|_2^2$ and the entropy functions $\psi(\mathbf{w}) = \sum_{i=1}^d w_i \log(w_i) - w_i$.

With a non-negative decreasing stepsize $\eta^t = \mathcal{O}(1/\sqrt{t})$, the DDA algorithm is known to converge at a rate of $\mathcal{O}(1/\sqrt{t})$ (or $\mathcal{O}(\sqrt{T})$ regret for online learning) to the optimal solution. It also establishes a tight connection between the convergence rate of distributed subgradient methods to the spectral properties $1 - \sigma_2(A)$ of the underlying graph.

D-ADMM: Its sub-linear convergence rate is later improved by the primal-dual solution in Decentralized ADMM (D-ADMM in [38]), where the authors reformulate the optimization as a constrained problem:

$$\begin{aligned} \min_{\{\mathbf{w}_i\}, \{\mathbf{z}_{i,j}\}} \quad & \sum_{i=1}^m f_i(\mathbf{w}_i), \\ \text{s.t.} \quad & \mathbf{w}_i = \mathbf{z}_{i,j} ; \mathbf{w}_j = \mathbf{z}_{i,j}, \quad \forall (i,j) \in A. \end{aligned}$$

The auxiliary variable $\mathbf{z}_{i,j}$ ensures parameters on the two sides of a connected edge are the same, which indeed guarantees consensus for the whole network. By concatenating

all $\mathbf{w}_i \in \mathbb{R}^d$ into $\mathbf{w} \in \mathbb{R}^{md}$, the optimization problem can be written in an ADMM algorithms:

$$\begin{aligned} \min_{\mathbf{w}, \mathbf{z}} \quad & f(\mathbf{w}) + g(\mathbf{z}), \\ \text{s.t.} \quad & A\mathbf{w} + B\mathbf{z} = 0. \end{aligned}$$

S-DCA: A similar idea of considering the optimization as a constrained problem can be found in [39], where the authors consider the constraints as $W\sqrt{L}$ with L denoting a positive symmetric semi-definite matrix (e.g., the Laplacian matrix). The original primal problem is now reformulated as a dual problem:

$$\max_{\lambda \in \mathbb{R}^{d \times n}} -f^*(\lambda\sqrt{L}).$$

2.3 Multi-task Optimization

Traditional offline multi-task learning refers to the setting when the learner has access to data from multiple related learning tasks and aim at improving the general performance by collaboration between the tasks. The first multi-task model is proposed in citeevgeniou2005learning, where the authors consider a regularized functional that encodes multi-task relations over tasks, thus biasing the solution of the problem towards functions that lie close to each other. Argyriou et al.[40] build on this formalization to simultaneously learn a multi-task classifier and the underlying spectral dependencies among tasks. A different approach is discussed in Ando and Zhang[41] where a structural risk minimization method is presented and multitask relations are established by enforcing predictive functions for the different tasks to belong to the same hypothesis set. Complexity results for multi-task learning under statistical assumptions are also given in Maurer (2006) [42].

In the context of online learning, multi-task problem is first studied in [43] within the learning with expert advice model. Specifically, in this model. the forecaster has access to a fixed set of experts and is expected to make predictions for K different tasks.

Regret bounds are given under the assumption that the set of best experts for the K tasks is small, as a way to formalize task similarity. Whereas these studies consider a multi-task protocol in which a single task is acted upon at each time step (what we call in this thesis the adversarially chosen task protocol), the work of Lugosi et al. [44] investigates the problem where an action for each task must be chosen at every step. The relatedness among tasks is captured by imposing restrictions on the joint action chosen at each step.

OMTL: Cavallanti and Cesa [45] introduce a new perceptron-based algorithm for the online multi-task binary classification problem. They allow a single task to be updated while other similar ones can share their updating with a pre-defined interaction matrix A . The general interaction matrix can be:

$$A^{-1} = \frac{1}{(1+b)K} \begin{bmatrix} b+K & b & \dots & b \\ b & b+K & \dots & b \\ \dots & \dots & \dots & \dots \\ b & \dots & \dots & b+K \end{bmatrix}$$

Their multi-task Perceptron rules can be simplified as,

$$w_{j,t} = w_{j,t-1} + \begin{cases} \frac{b+K}{(1+b)K} y_t x_t & \text{if } j = i_t \\ \frac{b}{(1+b)K} y_t x_t & \text{otherwise} \end{cases}$$

By using the same idea as kernel Perceptron bound [23], the number of mistakes m made by their algorithm is no more than

$$m \leq \sum_{t \in M} l_t(u) + \max_{i=1, \dots, K} (A)_{i,i}^{-1} (u^T A_{\otimes} u) + \sqrt{\max_{i=1, \dots, K} (A)_{i,i}^{-1} (u^T A_{\otimes} u) \sum_{t \in M} l_t(u)}$$

OTRL: Later Saha and Rai [46] point out that assuming a fixed interaction matrix and forcing all the tasks to be positively related will not always be the right choice since their inherent interactions may vary from time to time. Therefore, they propose a new model by allowing the interaction matrix to be adaptive with the data. They formulate

their optimization problem as:

$$\min_{\mathbf{w} \in \mathbb{R}^{kd}, \mathbf{A} > 0} \left[D_w(\mathbf{w} || \mathbf{w}_s) + D_A(\mathbf{A} || \mathbf{A}_s) + \sum_1^t l_t(\mathbf{w}) \right]$$

where D denotes Bregman Divergence $D_\phi(\mathbf{X}, \mathbf{Y}) = \phi(\mathbf{X}) - \phi(\mathbf{Y}) + \text{tr}((\mathbf{X} - \mathbf{Y})\mathbf{f}(\mathbf{Y})^T)$. They consider the following matrix divergences by substituting the appropriate function for ϕ

1. **LogDet Divergence:** For the LogDet matrix divergence, $D_{\phi_{LD}}(X, Y) = \text{tr}(XY^{-1}) - \log |XY^{-1}| - n$, and

$$\mathbf{A}_s = (\mathbf{A}_{s-1}^{-1} + \eta \text{sym}(\mathbf{W}_{s-1}^T \mathbf{W}_{s-1}))^{-1}$$

2. **Von-Neumann Divergence:** For the Von matrix divergence, $D_{\phi_{VN}}(X, Y) = \text{tr}(X \log(X) - Y \log(Y) - X + Y)$, and

$$\mathbf{A}_s = \exp(\log \mathbf{A}_{s-1} - \eta \text{sym}(\mathbf{W}_{s-1}^T \mathbf{W}_{s-1}))$$

3. **Covariance:** For Covariance model, the optimal task relationship matrix can be

$$\mathbf{A}_s = \frac{(\mathbf{W}_{s-1}^T \mathbf{W}_{s-1})^{\frac{1}{2}}}{\text{tr}((\mathbf{W}_{s-1}^T \mathbf{W}_{s-1})^{\frac{1}{2}})}$$

For more studies on (online) multi-task learning, the reader is referred to the survey in [47].

Chapter 3

Online Federated Learning with Edge Sampling

The first problem we address for online federated learning (OFL) in this thesis is designing communication-efficient algorithms with edge sampling technique. In the previous chapters, we point out the performance of OFL is affected by the mixing ability of the underlying graph [48–50], which is often quantified by the corresponding spectral gap $1 - \sigma_2(A)$ defined by the combination matrix A on the graph. For practical concerns, well-connected networks (*e.g.*, complete graphs) are naturally preferred for machine learning problems since it guarantees sufficient information exchange on each round and leads to faster convergence rates. However, on the other side of the coin is the heavy computation and communication cost when combining neighborhood parameters. For instance, OFL on a complete network with n nodes requires each node to combine neighborhood variables with $\mathcal{O}(nd)$ cost in combination, which clearly dominates the following $\mathcal{O}(d)$ cost in the local adaptation phase if the projection can be efficiently computed.

Such a mass of neighboring weights flooding into a certain node simultaneously may also lead to computation burdens beyond each node’s capacity. As a canonical example, wireless sensor networks (WSN) often use thousands of sensors to detect signals while each sensor may have only limited computation and storage ability for cost

concerns [51]. In the meanwhile, communication expenditure is often the primary concern of distributed algorithms since external communication directly relates to network bandwidth usage, energy consumption and overall running time [52, 53].

These practical challenges lead to a natural question: Can we design algorithms to alleviate the communication cost for dense networks while still maintaining a comparable convergence rate? In this chapter, we provide a solution by replacing the heavy deterministic communication with a stochastic communication strategy based on a novel sampling technique named ‘edge sampling’. We design a graph-dependent sampling rate to avoid the redundant communication on dense networks while sampling more frequently on poor-connected networks. Our algorithm enjoys the merits as follows: (1) alleviating the communication burden on each round ; (2) allowing the network to reach the consensus with the minimal communication cost. As a particular example, we show for the n -complete network our proposed algorithm obtains an $\mathcal{O}(n)$ times faster convergence rate than the unsampled algorithm if given the same communication budget, which is further confirmed by numerous experiments on various graphs.

3.1 Existing Solutions

Existing solutions: The idea of replacing the deterministic combination with a stochastic counterpart to reduce communication burden on each round is straightforward, where each node exchanges information with only a few of its immediate neighbors. The earliest work (Gossip algorithm [50]) presents a simple scheme by requiring one node to wake up with a certain probability and averaging its value with one random neighbor. Despite its broad success in asynchronous settings, the requirement of disjoint communication is often hard to be satisfied in synchronous settings. To amend this issue, the authors design a synchronous gossip algorithm by separating all nodes into two categories with equal probability, namely ‘active group’ and ‘inactive group’. On each round, only ‘active’ node can select one neighbor to communicate, and all the ‘inactive’ nodes stay still and wait for communication. If the selected neighbor of the node

happens to be ‘inactive’ and has not been selected by other nodes, the mutual communication is executed. However, rejection often happens since its selected neighbor may also be ‘active’ or has been sampled by other ‘active’ neighbors.

The following research in [37] extends this work by forming time-dependent degree matrix $D(t)$ and adjacency matrix $C(t)$, and updates weights through the newly-formed Laplacian Matrix. These solutions, which we shall name as *node sampling*, generate a biased estimation of the combination step and often suffer from large sampling variance due to the low sampling rate involved. To summarize, decentralized optimization with graph sampling is still an area that has not been fully explored.

Our Solution: Our work is motivated by the previous node sampling strategy, but we focus on proposing and analyzing a communication-efficient algorithm based on a new sampling strategy named ‘edge sampling’. More specifically, each node selects a subset of its connecting edges based on a sampling parameter, as well as the original combination weight $a_{i,j}$, to generate an unbiased estimation when combining neighboring parameters. Different from the universal method in node sampling, our sampling strategy is graph-dependent as we seek for the optimal sampling rate for each individual graph, which allows us to use a relatively small sampling rate to avoid the redundant communication on dense networks while still keep most of the connections on sparse graphs. Our sampling strategy is also unbiased with a lower variance when compared with previous biased estimation. A full comparison of all algorithms is listed in Table 3.1.

	Unbiased Estimation	Low Variance	Graph Dependent	Communication Efficient
DDA [37]	✓	✓	✓	×
Synchronous Gossip [50]	×	×	×	✓
DDA-NS [37]	×	×	×	✓
Our Algorithm	✓	✓	✓	✓

TABLE 3.1: Algorithm Comparison

3.2 Online Federated Learning with Edge Sampling

We formally present our online federated learning based on the edge sampling technique to achieve the fastest convergence rate with the same communication budget in this part. Although this algorithm can be generally applied to most decentralized algorithms, we focus on the broadly-used ‘Distributed Dual Averaging algorithm’ (DDA) [37] in this part.

3.2.1 Sampling Strategy

The edge sampling algorithm, named as DDA-ES (2), requires each existing edge to appear with probability

$$p_{i,j} = \frac{1}{1 + \frac{z}{a_{i,j}}}, \quad z > 0, \quad (3.1)$$

with z denoting a graph-dependent parameter to control the sampling rate. For each edge $(i, j) \in E$, its sampling rate $p_{i,j}$ is set to be positively correlated to $a_{i,j}$ since the combination weight $a_{i,j}$ between node i and j often indicates the importance of a certain edge $(i, j) \in E$. For instance, if the edge acts as a bottleneck of the underlying graph, the combination weight $a_{i,j}$ for this edge is generally larger to increase information exchange, and the edge sampling strategy also samples this edge more frequently to boost communication between nodes.

When an edge $(i, j) \in E$ is sampled on round t , the new combination weights $b_{i,j}^t$ and $b_{j,i}^t$ will be scaled by a factor $1/p_{i,j}$ to provide an unbiased estimation, namely

$$b_{i,j}^t = b_{j,i}^t = \begin{cases} \frac{1}{p_{i,j}} \cdot a_{i,j} & \text{if } e_{i,j} \text{ appears;} \\ 0 & \text{otherwise,} \end{cases}$$

and

$$\mathbb{E} [b_{i,j}^t] = p_{i,j} \cdot \frac{1}{p_{i,j}} a_{i,j} = a_{i,j}.$$

Algorithm 2 Distributed Dual Averaging with Edge Sampling (**DDA-ES**)

-
- 1: **Input:** Convex set Ω , combination matrix A , step-size $\{\eta^t\}$ for $\forall i \in \{1, \dots, n\}$.
 - 2: **Initialize:** $\mathbf{w}_i^0 = \mathbf{0}$ and $\mathbf{z}_i^0 = \mathbf{0}$ for $\forall i \in \{1, \dots, m\}$.
 - 3: **Set** $0 \leq z \leq \min_{a_{i,j} \neq 0} \left\{ \frac{[A^2]_{i,j}}{2a_{i,j}} \right\}$
 - 4: **for** $t = 1, \dots, T$ **do**
 - 5: Sample each edge with probability $p_{i,j} = \frac{1}{1 + \frac{z}{a_{i,j}}}$.
 - 6: **if** edge $e_{i,j}$ appears **then**
 - 7: Assign weight $b_{i,j}^t = b_{j,i}^t = \frac{1}{p_{i,j}} a_{i,j}$
 - 8: **else**
 - 9: Assign weight $b_{i,j}^t = b_{j,i}^t = 0$
 - 10: **end if**
 - 11: Normalize: $b_{i,i}^t = 1 - \sum_{j=1 \setminus i}^n b_{i,j}^t$ for $i \in [1, \dots, n]$.
 - 12: **for** Each learner $i \in V$ **do**
 - 13: Combine: $\mathbf{z}_i^{t+1} = \sum_{j \in N(i)} b_{i,j}^t \mathbf{z}_j^t + \partial f_i(\mathbf{w}_i^t)$
 - 14: Adapt: $\mathbf{w}_i^{t+1} = \Pi_{\Omega}^{\psi}(\mathbf{z}_i^{t+1}, \eta^t)$
 - 15: **end for**
 - 16: **end for**
 - 17: **Output:** $\mathbf{w}_i^T$ for $i = 1, \dots, n$
-

A normalization step is finally performed on each node to preserve row-stochasticity for matrix B :

$$b_{i,i}^t = 1 - \sum_{j=1 \setminus i}^n b_{i,j}^t \quad \text{for } i \in [1, \dots, n].$$

It's worth mentioning that although $b_{i,i}^t$ also acts as an unbiased estimation for $a_{i,i}$, the new combination matrix $B(t)$ is not strictly 'doubly stochastic' due to the fact that $b_{i,i}^t$ can occasionally be negative. However, we shall show the bound for z

$$0 \leq z \leq \min_{a_{i,j} \neq 0} \left\{ \frac{[A^2]_{i,j}}{2a_{i,j}} \right\} \quad (3.2)$$

is sufficient to ensure the variance matrix $\mathbb{E}[B^2(t)]$ to be non-negative and therefore guarantee the convergence of our proposed algorithm in the appendix.

3.2.2 Comparing Node Sampling with Edge Sampling

Based on the updating strategy, we can notice that the above edge sampling algorithm differs from existing node sampling algorithms [37, 50] in the following aspects:

1. Node sampling algorithms adopt a ‘lazy random walk’ scheme and generate a biased estimation due to the low sampling rate. For instance, the expected communication matrix in [37] is $\mathbb{E}[B(t)] = \left(\frac{d_{max}}{d_{max}+1}\right)^2 I + \frac{2d_{max}+1}{(d_{max}+1)^2} A$, while edge sampling strictly generates an unbiased estimation $\mathbb{E}[B(t)] = A$.
2. Node sampling algorithms assume one node to interact with at most one neighbor on each round, while multiple communication can happen to a certain node in edge sampling algorithm, leading to a lower sampling variance.
3. Node sampling algorithms adopt a fixed sampling strategy for all graphs, which the sampling rate in edge sampling is graph-dependent.

In the following theoretical analysis, we shall show these merits lead to a more faster convergence rate when given the same communication budget.

3.2.3 Implementations of Edge Sampling

Clearly, the above edge sampling algorithm cannot be directly executed since most decentralized algorithms are implemented on computation nodes, and we shall provide its node-wise implementation as follows.

For each edge $(i, j) \in E$, node i and node j can trigger the interaction on $(i, j) \in E$ independently with probability $\hat{p}_{i,j} = \hat{p}_{j,i} = 1 - \sqrt{1 - p_{i,j}}$. Any node will respond to its neighbor’s communication requirement and scale the combination weight to $b_{i,j}$. Therefore, on each round, the communication between node i and node j is executed with probability $1 - (1 - p_i)(1 - p_j) = p_{i,j}$.

3.3 Convergence Analysis

We now turn to the convergence analysis for our proposed edge sampling algorithm, along with some discussions on the sampling strategy for different graphs.

3.3.1 Convergence Analysis

The following theorem provides a general convergence rate for any connected graphs:

Theorem 3.1. *Let $\{B(t)\}$ be an i.i.d. sequence of doubly stochastic matrices generated in Algorithm 2. By setting the step size $\eta^t \propto \frac{R\sqrt{1-\kappa(z)}}{L\sqrt{t}}$, we have*

$$f(\hat{\mathbf{w}}_i^T) - f(\mathbf{w}^*) \leq c \frac{RL}{\sqrt{T}} \frac{\log(T\sqrt{n})}{\sqrt{1-\kappa(z)}} \quad (3.3)$$

holds with probability at least $1 - 1/(T^2n)$ for any node i , where $\psi(\mathbf{w}^*) \leq R^2$, c is a universal constant, $\kappa(z) = \max \{ \lambda_2^2(A) - 2z\lambda_2(A) + 2z, \lambda_n^2(A) - 2z\lambda_n(A) + 2z \}$ and $\hat{\mathbf{w}}_i^T$ denotes the average weight $(\mathbf{w}_i^1 + \dots + \mathbf{w}_i^T)/T$.

The above convergence rate relies on two extreme eigenvalues $\lambda_2(A)$ and $\lambda_n(A)$ that control the mixing rate of the underlying graph, and is consistent with previous research [33, 34, 54]. Its reliance on the sampling parameter z is not surprising, and we have to choose a proper z so that the gain in reducing the communication cost does not offset the loss in the convergence rate.

3.3.2 Communication Comparison

To compare communication efficiency, we recall the convergence rate for the standard DDA (Theorem 2, [37])

$$f(\hat{\mathbf{w}}_i^{T'}) - f(\mathbf{w}^*) \leq c' \frac{RL}{\sqrt{T'}} \frac{\log(T'\sqrt{n})}{\sqrt{1-\sigma_2(A)}}, \quad (3.4)$$

where $\sigma_2(A) = \max \{\lambda_2(A), |\lambda_n(A)|\}$ and c' is another universal constant. We shall focus on their convergence comparison by adopting a scheme proposed in [55] by fixing the communication budget.

Suppose given the communication budget B , the standard DDA algorithm is permitted to update T' rounds. Clearly, DDA-ES algorithm is able to update $T = T'/\bar{p}(z)$ rounds due to the low communication cost per iteration, with $\bar{p}(z)$ denoting the averaging sampling rate:

$$\bar{p}(z) := \frac{\sum_{i=1}^n \sum_{j=i+1}^n p_{i,j}}{|E|} = \frac{\sum_{i=1}^n \sum_{j=i+1}^n \frac{a_{i,j}}{a_{i,j}+z}}{|E|}.$$

Substitute T in (3.3) with $T'/\bar{p}(z)$, we have

$$\begin{aligned} f(\hat{\mathbf{w}}_i^{T'}) - f(\mathbf{w}^*) &\leq c \frac{RL}{\sqrt{T'/\bar{p}(z)}} \frac{\log(T'\sqrt{n}/\bar{p}(z))}{\sqrt{1-\kappa(z)}} \\ &= c \frac{RL}{\sqrt{T'}} \frac{\log(T'\sqrt{n}/\bar{p}(z))}{\sqrt{1-\sigma_2(A)}} \frac{\sqrt{\bar{p}(z)}\sqrt{1-\sigma_2(A)}}{\sqrt{1-\kappa(z)}}. \end{aligned} \quad (3.5)$$

For simplicity of analysis, we ignore the effect of constant values c, c' and assume T' is sufficiently large so that $\log(T'\sqrt{n})$ dominates $\log(1/\bar{p}(z))$, or $\log(T'\sqrt{n}/\bar{p}(z)) = \log(T'\sqrt{n}) + \log(1/\bar{p}(z)) \approx \log(T'\sqrt{n})$.

Therefore, the difference between the convergence rates of DDA (3.4) and DDA-ES (3.5) can be expressed by a factor:

$$\Gamma(z) := \sqrt{\bar{p}(z)} \cdot \frac{\sqrt{1-\sigma_2(A)}}{\sqrt{1-\kappa(z)}}. \quad (3.6)$$

As can be seen from above formulation, $\Gamma(z)$ is constituted by two components: the lower average sampling rate $\bar{p}(z)$ in DDA-ES clearly reduces $\Gamma(z)$ while the decreasing spectral gap $1 - \kappa(z)$ increases our target value on the contrary. Therefore, the optimal z^* corresponds to the best trade-off between these two components, namely

$$z^* = \arg \min_{z \in (3.2)} \Gamma(z) = \arg \min_{z \in (3.2)} \frac{\bar{p}(z)}{1 - \kappa(z)}. \quad (3.7)$$

3.3.3 Discussion on Graphs

For many graphs, Eq (3.7) has a closed-form solution and we can clearly see whether sampling accelerates the original algorithm.

3.3.3.1 N-complete graph

Our first example is the n-complete graph where each node is connected to all other nodes and therefore the graph admits the best connectivity.

Corollary 3.2. *For a n-complete graph with $A = \frac{\mathbf{1}_n \mathbf{1}_n^T}{n}$, the optimal z^* is $\frac{4}{n+2}$ and the corresponding $\Gamma(z)$ is $\mathcal{O}(1/\sqrt{n})$.*

Remark: (1) To achieve $f(\hat{\mathbf{w}}_i(T')) - f(\mathbf{w}^*) \leq \epsilon$, the original DDA algorithm needs approximately $\mathcal{O}(\frac{R^2 L^2}{\epsilon^2})$ communication rounds, while the edge sampling algorithm only needs approximately $\mathcal{O}(\frac{R^2 L^2}{n \epsilon^2})$ communication rounds. In other words, the sampling algorithm is approximately n -times faster than the original DDA when given the same communication budget.

(2) With $p_{i,j} = 4/(n+2)$, the communication cost for each node on one iteration is reduced from $\mathcal{O}(nd)$ in DDA to $\mathcal{O}(4d)$ in DDA-ES, which clearly reduces the communication cost for each node and avoids the network overburden problem. The price we pay is the half convergence rate on each round due to the reduced communication and the increase of variance.

(3) In DDA-NS [37], the average sampling rate is reduced to $\frac{2d_{max}+1}{(d_{max}+1)^2}$, while the spectral gap decreases by the same factor. Namely, $\Gamma = 1$ for their algorithm.

Proof: For n-completer graph, we can directly calculate $\lambda_2 = \lambda_n = 0$, $\kappa(z) = 2z$ and the requirement in Eq (3.2) becomes $0 \leq z \leq \frac{1}{2}$.

$$\begin{aligned} z^* &= \arg \min_z \frac{\bar{p}(z)}{1 - \kappa(z)} \\ &= \arg \min_z \frac{1}{1 + nz} \cdot \frac{1}{1 - z} \end{aligned}$$

$$= \frac{n-2}{4n}$$

Therefore, we have $p_{i,j} = \frac{4}{n+2}$ and the corresponding $\Gamma^2(z) \approx \frac{8}{n}$.

3.3.3.2 D-regular Graph

We then consider the d-regular graph, which is known to be a well-connected graph when d is sufficiently large. Recall our average sampling rate on the d-regular graph is

$$\bar{p}(z) = \frac{\sum_{i=1}^n \sum_{j=i+1}^n p_{i,j}}{|E|} = \frac{1}{1+dz}.$$

(1) When $\lambda_2 + 2\lambda_n \geq 1$, we can set $z = \frac{1+\lambda_2}{4}$. Clearly, $z \leq \frac{\lambda_2+\lambda_n}{2}$ and $\kappa(z) = \lambda_2^2 - 2z\lambda_2 + 2z$. We have

$$\begin{aligned} \Gamma^2(z) &= \frac{1/d}{1/d+z} \cdot \frac{1-\lambda_2}{1-\lambda_2^2+2z\lambda_2-2z} \\ &= \frac{1}{1+d \cdot \frac{1+\lambda_2}{4}} \cdot \frac{1}{1+\lambda_2-2 \cdot \frac{1+\lambda_2}{4}} \\ &\approx \frac{8}{(1+\lambda_2)^2} \cdot \frac{1}{d} \end{aligned}$$

(2) When $2\lambda_2 + \lambda_n \leq 1$, we can set $z = \frac{1+\lambda_n}{4}$. Clearly, $z \geq \frac{\lambda_2+\lambda_n}{2}$, and $\kappa(z) = \lambda_n^2 - 2z\lambda_n + 2z$. We have

$$\begin{aligned} \Gamma^2(z) &= \frac{1/d}{1/d+z} \cdot \frac{1-\lambda_2}{1-\lambda_n^2+2z\lambda_n-2z} \\ &= \frac{1}{1+d \cdot \frac{1+\lambda_n}{4}} \cdot \frac{1}{1+\lambda_n-2 \cdot \frac{1+\lambda_n}{4}} \cdot \frac{1-\lambda_2}{1-\lambda_n} \\ &\approx \frac{8}{(1+\lambda_n)^2} \cdot \frac{1-\lambda_2}{1-\lambda_n} \cdot \frac{1}{d} \end{aligned}$$

(3) When $\lambda_2 + 2\lambda_n \leq 1$ and $2\lambda_2 + \lambda_n \geq 1$, we can set $z = \frac{\lambda_2+\lambda_n}{2}$, and $\kappa(z) = \lambda_2^2 - 2z\lambda_2 + 2z$. We have

$$\Gamma^2(z) = \frac{1/d}{1/d+z} \cdot \frac{1-\lambda_2}{1-\lambda_2^2+2z\lambda_2-2z}$$

$$\begin{aligned}
&= \frac{1}{1 + d \cdot \frac{\lambda_n + \lambda_2}{2}} \cdot \frac{1}{1 + \lambda_2 - 2 \cdot \frac{\lambda_n + \lambda_2}{2}} \\
&= \frac{2}{2 + d(\lambda_n + \lambda_2)} \cdot \frac{1}{1 - \lambda_n}
\end{aligned}$$

For a large d , the d -regular graph represents a well-connected graph with a large spectral gap $1 - \sigma_2(A)$, namely λ_2 is sufficiently far from 1 and λ_n is sufficiently far from -1 . For our experiments with 75-connected neighbor on 1000-node cycle, $\lambda_2 + \lambda_n$ is also sufficient far from $[-1/d, 1/d]$, and therefore, the leading order of $\Gamma^2(z)$ is $\mathcal{O}(1/d)$ and we expect $\mathcal{O}(d)$ acceleration when given the same communication budget.

3.3.3.3 Star Graph

We also consider the star graph where there exists a center node and all other nodes will connect only to the center node. Such a graph is commonly used in centralized distributed optimization. We consider the communication matrix $A = I - \frac{1}{n}(D - J)$, with D denoting the degree matrix and J denoting the adjacency matrix. Without loss of generality, we mark the center node as node 1.

The spectrum of the star graph is known to be $\{1, 1 - \frac{1}{n}, 0\}$. Clearly, we have the average sampling rate as

$$\bar{p}(z) = \frac{\sum_{i=1}^n \sum_{j=i+1}^n p_{i,j}}{|E|} = \frac{1}{1 + nz}.$$

The requirement in Eq 3.2 becomes

$$\begin{aligned}
z &\leq \min_{a_{i,j} \neq 0} \left\{ \frac{[A^2]_{i,j}}{2a_{i,j}} \right\} \\
&= \frac{\sum_{k=1}^n a_{i,k} a_{k,1}}{2a_{i,1}} \\
&= \frac{a_{i,i} a_{i,1} + a_{i,1} a_{1,1}}{2a_{i,1}} \\
&= \frac{a_{i,i} + a_{1,1}}{2} \\
&= \frac{1}{2}
\end{aligned}$$

Setting $z = \frac{1}{2} - \frac{3}{4n}$ leads to $\Gamma^2(z) \approx \frac{2}{n}$.

3.3.3.4 General Case

In the general case, Eq (3.7) may not have a closed-form solution for an arbitrary graph, and we can use grid-search in $\left[0, \min_{a_{i,j} \neq 0} \left\{ \frac{[A^2]_{i,j}}{2a_{i,j}} \right\} \right]$ to find a (sub)optimal z with $\mathcal{O}(n^2)$ computation cost, or simply use $\min_{a_{i,j} \neq 0} \left\{ \frac{[A^2]_{i,j}}{2a_{i,j}} \right\}$ for convenience.

3.4 Experiments

To validate our theoretical results, we now report experimental results of the proposed edge sampling algorithm, followed by a series of discussion.

3.4.1 Experimental Setup

Competing Algorithms We compare our *DDA-ES* algorithm with the original *DDA* algorithm, as well as two existing node sampling algorithms, *Syn Gossip* [50] and *DDA-NS* [37]. Since communication rejections exist in *Syn Gossip* algorithm, we shall count for 1 communication if the exchange is executed, and count for 0.1 communication if its requirement is rejected, since each node may only need a minimal amount of bits to receive the decision from its target neighbor.

Graphs To investigate the efficiency of sampling on different networks, we conduct experiments on two types of graphs:

1. *k-connected cycle* $C_{n,k}$. The k -connected cycle represents a graph by placing n nodes on a cycle and connecting each node to its k neighbors on the left and right [37]. This represents a $(2k + 1)$ -regular graph with a closed-form solution for z , and the theoretical analysis (S.M. C2) indicates that *DDA-ES* should roughly achieve $\mathcal{O}(2k + 1)$ faster convergence rate than *DDA* when given the same communication budget.

2. *Random graph $G_{n,c}$.* The binomial random graph, also known as the ‘Erdos-Renyi model’ [56], consists of vertex set $V = \{1, 2, \dots, n\}$ in which any two nodes are connected with probability c . This represents a general graph without closed-form solutions for z . We set the connection ability as $c = 0.15$ and use grid-search to find $z^* = 0.05$.

Parameter Settings For all algorithms, we set $\eta^t = \mathcal{O}(1/\sqrt{t})$ as suggested by previous theoretical analysis. The communication matrix A is set by the ‘Metropolis-Hasting’ rule [57, 58] to guarantee doubly stochasticity:

$$A_{i,j} = \begin{cases} \min \{1/d(i), 1/d(j)\} & \text{if } (i, j) \in E \text{ and } i \neq j; \\ \sum_{k \in N(i)} \max \{0, 1/d(i) - 1/d(k)\} & \text{if } i = j; \end{cases}$$

3.4.2 Mixing Rate of Markov Chain

The mixing rate of Markov chain is known to reflect the diffusion ability of the underlying graph [49], and also plays an important role in analyzing the convergence rate of distributed algorithms [37]. We estimate the mixing rate of Markov chain based on the communication matrix A as follows. Each node is given an initial value w_i^0 based on a predefined n -dimensional simplex and exchanges their information as $w_i^{t+1} = \sum_{j \in N(i)} a_{i,j} w_j^t$. We measure the difference between $W^t = (w_0^t, \dots, w_n^t)$ and the optimal distribution $\mathbf{1}_n/n$ by calculating their gap, namely $\|W^t - \mathbf{1}_n/n\|_2$.

We first report the mixing rate of Markov chain on k -connected cycle, which measures the diffusion ability of our sampling strategy. Based on figure 3.1, the edge sampling algorithm only needs a few rounds to reach its equilibrium distribution $\mathbf{1}_n/n$ and significantly outperforms the unsampled algorithm. This implies that our sampling algorithm can efficiently diffuse information to its neighborhood, which also illustrates the excellent agreement of the empirical behavior with our theoretical predictions. Its sampling superiority over node sampling algorithms can also be directly observed, and similar phenomenon holds on the random graph.

Figure 3.1: Mixing rate of Markov chain on different graphs with the gap referring to $\|W^t - \mathbf{1}_n/n\|_2$. Left is the result on the k -connected cycle with $n = 1000$, $k = 75$ and $z = 0.25$; right is the result on the random graph with $n = 1000$, $c = 0.15$ and $z = 0.05$.

3.4.3 Binary Classification

We consider a distributed binary classification problem based on the logistic regression function

$$f_i(\mathbf{w}) = \log \left(1 + e^{-y_i \cdot \mathbf{w}^\top \mathbf{x}_i} \right)$$

on *a9a* dataset¹. Each node on the graph receives a subset of the dataset, and computes its own parameters based on the loss function, followed by communication with its neighbors. The proximal function is set as $\psi(\mathbf{w}) = \frac{\lambda}{2} \|\mathbf{w}\|^2$ with $\lambda = 10^{-3}$. We measure the performance of all algorithms by using the weight on the first node to calculate the gap $f(\hat{\mathbf{w}}_1^{T'}) - f(\mathbf{w}^*)$. Each *iteration round* refers to a single execution of combination and adaption, and each *communication round* refers to the standard communication cost for one iteration in DDA algorithm.

Figure 3.2 illustrates the convergence rate in terms of each iteration round. As can be observed from the figures, the standard DDA algorithm has the fastest convergence rate since it combines information from all neighbors. With a rather low sampling rate, the DDA-ES algorithm converges almost exactly like the standard DDA, which clearly validates its communication efficiency. On the other hand, both Syn Gossip and DDA-NS algorithms cannot match the performance of these two algorithms. In particular, due to the lowest sampling rate, $f(\hat{\mathbf{w}}_1^{T'}) - f(\mathbf{w}^*)$ for Syn Gossip algorithm only occasionally drops down when information exchange is executed, while for most of the time, it only

¹<https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>

Figure 3.2: Distributed binary classification on different graphs with the gap referring to $f(\hat{\mathbf{w}}_1^{T'}) - f(\mathbf{w}^*)$. Left is the result on the k -connected cycle with $n = 1000$, $k = 75$ and $z = 0.25$; right is the result on the random graph with $n = 1000$, $c = 0.15$ and $z = 0.05$.

minimizes towards its own local function and therefore performs poorly on the overall datasets.

Figure 3.2 also compares the communication-efficiency for all four algorithms. Consistent with previous Markov chain experiments, DDA-ES enjoys the best communication utilization and achieves the fastest convergence rate when given the same communication budget. We can also observe DDA-ES algorithm suffers from a lower variance when compared with Syn Gossip and DDA-ES algorithms.

Specially, the left top figure in Figure 3.2 implies DDA-ES achieves similar performance of DDA with only $p \approx \mathcal{O}(4/(2k + 1))$ communication cost on each iteration, we can expect it to converge $\mathcal{O}((2k + 1)/4)$ faster when given the same communication budget, which clearly shows a consistent result with our previous theoretical analysis.

3.5 Budget-Efficient Decentralized Optimization

In our method, we assume the local adaptation is relatively cheap compared with external communication, and therefore the sampling algorithm is allowed to update more iterations when given a fixed communication cost. In case the local computation is comparable to communication, we consider a budget-efficient decentralized optimization as follows.

Suppose the expenditure for each local adaptation is c_1 and the cost for external communication is c_2 . For a given budget B , the DDA is allowed to update T' rounds, namely $B = (c_1 + c_2)T'$. Then with the same budget, the edge sampling algorithm can perform

$$\frac{B}{c_1 + c_2 \hat{p}(z)} = \frac{c_1 + c_2}{c_1 + c_2 \hat{p}(z)} T'$$

iterations. Replacing T in (6) with the above formulation and we can have a similar analysis to calculate the optimal z^* .

3.6 Chapter Summary

In this chapter, we propose and analyze a decentralized optimization algorithm with stochastic communication based on a new sampling strategy named ‘edge sampling’. The new sampling strategy provides an unbiased i.i.d. estimation of the initial combination matrix, but the communication cost per round is significantly reduced on dense and well-connected networks while a comparable convergence rate is still preserved.

When compared with existing node sampling algorithms, our strategy shows its superiorities for suffering from less sampling variance and being more communication-efficient, which are further validated by both theoretical analysis for its convergence rate and numerical experiments on the mixing rates of Markov chain and distributed machine learning problems.

3.7 Appendix: Proof for Theorem 3.1

Theorem 3.1. *Let $\{B(t)\}$ be an i.i.d. sequence of doubly stochastic matrices generated in Algorithm 2. By setting the step size $\eta^t \propto \frac{R\sqrt{1-\kappa(z)}}{L\sqrt{t}}$, we have*

$$f(\hat{\mathbf{w}}_i^T) - f(\mathbf{w}^*) \leq c \frac{RL}{\sqrt{T}} \frac{\log(T\sqrt{n})}{\sqrt{1-\kappa(z)}} \quad (3.3)$$

holds with probability at least $1 - 1/(T^2n)$ for any node i , where $\psi(\mathbf{w}^) \leq R^2$, c is a universal constant, $\kappa(z) = \max \{ \lambda_2^2(A) - 2z\lambda_2(A) + 2z, \lambda_n^2(A) - 2z\lambda_n(A) + 2z \}$ and $\hat{\mathbf{w}}_i^T$ denotes the average weight $(\mathbf{w}_i^1 + \dots + \mathbf{w}_i^T)/T$.*

To prove Theorem 3.1, we first derive the formulation of the variance matrix $\mathbb{E}[B^2]$ in the following lemma.

Lemma 3.3. *For edge sampling algorithm, the variance matrix $\mathbb{E}[B(t)^2]$ can be derived as $\mathbb{E}[B(t)^2] = A^2 - 2z \cdot A + 2z \cdot I$, and the condition $0 \leq z \leq \min_{a_{i,j} \neq 0} \left\{ \frac{[A^2]_{i,j}}{2a_{i,j}} \right\}$ is sufficient to guarantee $\mathbb{E}[B(t)^2]$ to be doubly stochastic.*

Remark: Since $\mathbb{E}[B(t)^2]$ is guaranteed to be doubly-stochastic, its first (also the largest) eigenvalue is 1 and the corresponding first eigenvector is $\mathbf{1}_n/n$. In the meanwhile, we can derive its second largest eigenvalue as

$$\lambda_2(\mathbb{E}[B(t)^2]) = \kappa(z) = \max \{ \lambda_2^2(A) - 2z\lambda_2(A) + 2z, \lambda_n^2(A) - 2z\lambda_n(A) + 2z \},$$

which controls the mixing rate of the sampling graph in the later convergence analysis.

Proof: We start from calculating the element-wise variance as follows:

1) For $m \neq i, j$, (i, m) and (m, j) represents two independent edges, and we have

$$\mathbb{E}[b_{i,m}b_{m,j}] = a_{i,m}a_{m,j}$$

2) If $m = i$, then

$$\begin{aligned}
\mathbb{E}[b_{i,i}b_{i,j}] &= \mathbb{E}\left[\left(1 - \sum_{k \setminus i} b_{i,k}\right) b_{i,j}\right] \\
&= \mathbb{E}[b_{i,j}] - \mathbb{E}\left[\sum_{k \setminus i,j} b_{i,k}b_{i,j}\right] - \mathbb{E}[b_{i,j}^2] \\
&= a_{i,j} - \sum_{k \setminus i,j} a_{i,k}a_{i,j} - \frac{1}{p_{i,j}}a_{i,j}^2 \\
&= a_{i,j}a_{i,i} + \left(1 - \frac{1}{p_{i,j}}\right)a_{i,j}^2 \\
&\stackrel{\textcircled{1}}{=} a_{i,j}a_{i,i} - a_{i,j}z,
\end{aligned}$$

where $\textcircled{1}$ uses $p_{i,j} = \frac{1}{1 + \frac{z}{a_{i,j}}}$.

3) For diagonal elements:

$$\begin{aligned}
\mathbb{E}[b_{i,i}b_{i,i}] &= \mathbb{E}\left[\left(1 - \sum_{k \setminus i} b_{i,k}\right)^2\right] \\
&= 1 - \mathbb{E}\left[2 \sum_{k \setminus i} b_{i,k}\right] + \mathbb{E}\left[\left(\sum_{k \setminus i} b_{i,k}\right)^2\right] \\
&= 1 - 2 \sum_{k \setminus i} a_{i,k} + \sum_{k \setminus i} \frac{1}{p_{i,k}}a_{i,k}^2 + \sum_{k \setminus i} \sum_{k' \setminus i, k} a_{i,k}a_{i,k'} \\
&= 1 - 2 \sum_{k \setminus i} a_{i,k} + \left(\sum_{k \setminus i} a_{i,k}\right)^2 + \sum_{k \setminus i} \left(\frac{1}{p_{i,k}} - 1\right)a_{i,k}^2 \\
&= a_{i,i}^2 + \sum_{k \setminus i} \left(\frac{1}{p_{i,k}} - 1\right)a_{i,k}^2 \\
&\stackrel{\textcircled{1}}{=} a_{i,i}^2 + \sum_{k \setminus i} a_{i,k}z \\
&\stackrel{\textcircled{2}}{=} a_{i,i}^2 + z - a_{i,i}z,
\end{aligned}$$

where $\textcircled{1}$ uses $p_{i,j} = \frac{1}{1 + \frac{z}{a_{i,j}}}$ and $\textcircled{2}$ uses $\sum_{k=1} a_{i,k} = 1$.

With these preparations in place, we can calculate $\mathbb{E}[B^2]_{i,j}$ as follows:

(1) For off-diagonal elements:

$$\begin{aligned}
\mathbb{E} [B(t)^2]_{i,j} &= \mathbb{E} \left[\sum_k b_{i,k} b_{k,j} \right] \\
&= \mathbb{E} \left[\sum_{k \setminus i,j} b_{i,k} b_{k,j} \right] + \mathbb{E} [b_{i,i} b_{i,j}] + \mathbb{E} [b_{i,j} b_{j,j}] \\
&= \sum_{k \setminus i,j} a_{i,k} a_{k,j} + a_{i,i} a_{i,j} - a_{i,j} z + a_{j,j} a_{i,j} - a_{i,j} z \\
&= \sum_k a_{i,k} a_{k,j} - 2a_{i,j} z \\
&= [A^2]_{i,j} - 2a_{i,j} z
\end{aligned}$$

The constraint $z \leq \min_{a_{i,j} \neq 0} \left\{ \frac{[A^2]_{i,j}}{2a_{i,j}} \right\}$ guarantees $\mathbb{E} [B(t)^2]_{i,j} \geq 0$.

Recall that we only sample on existing edges, and

$$[A^2]_{i,j} = \sum_k a_{i,k} a_{k,j} \geq (a_{i,i} + a_{j,j}) a_{i,j}$$

So a sufficient requirement to guarantee $z > 0$ will be $a_{i,i} \neq 0$ is true for all i , which can be easily satisfied since DDA algorithm needs to combine its own dual variables.

(2) For diagonal elements:

$$\begin{aligned}
\mathbb{E} [B(t)^2]_{i,i} &= \mathbb{E} \left[\sum_k b_{i,k} b_{k,i} \right] \\
&= \mathbb{E} \left[\sum_{k \setminus i} b_{i,k}^2 + b_{i,i}^2 \right] \\
&= \sum_{k \setminus i} \frac{1}{p_{i,k}} a_{i,k}^2 + a_{i,i}^2 + z - a_{i,i} z \\
&= \sum_k a_{i,k}^2 + \sum_{k \setminus i} \left(\frac{1}{p_{i,k}} - 1 \right) a_{i,k}^2 + z - a_{i,i} z \\
&= \sum_k a_{i,k}^2 + 2z - 2z a_{i,i} \\
&= [A^2]_{i,i} + 2z - 2z a_{i,i}
\end{aligned}$$

$\mathbb{E}[B(t)^2]_{i,i}$ is apparently non-negative.

These formulations of $\mathbb{E}[B(t)^2]_{i,j}$ and $\mathbb{E}[B(t)^2]_{i,i}$ can be concluded with one formula:

$$\mathbb{E}[B(t)^2] = A^2 - 2z \cdot A + 2z \cdot I$$

Notice that the row sums and column sums of $\mathbb{E}[B(t)^2]$ are all 1, and all its components are guaranteed to be non-negative, we come to the fact that $\mathbb{E}[B(t)^2]$ is doubly-stochastic. $\square$

Equipped with this lemma, we can now approach the convergence analysis for our proposed edge sampling algorithm by first reviewing several existing results that act as the foundations for our later analysis. In [37], the following convergence rate for standard DDA is established as:

$$f(\hat{\mathbf{w}}_i^T) - f(\mathbf{w}^*) \leq \frac{\psi(\mathbf{w}^*)}{T\eta^T} + \frac{L^2}{2T} \sum_{t=1}^T \eta^{t-1} + \frac{2L}{nT} \sum_{t=1}^T \sum_{j=1}^n \eta^t \|\bar{\mathbf{z}}^t - \mathbf{z}_j^t\|_* + \frac{L}{T} \sum_{t=1}^T \eta^t \|\bar{\mathbf{z}}^t - \mathbf{z}_i^t\|_*, \quad (3.8)$$

with $\bar{\mathbf{z}}^t = \frac{1}{n} \sum_{j=1}^n \mathbf{z}_j^t$ and $\hat{\mathbf{w}}_i^T = \frac{1}{T} \sum_{t=1}^T \mathbf{w}_i^t$.

The first two terms in the upper bound are optimization error terms that are common to centralized subgradient algorithms, and the last two terms are graph-related penalties depending on the global variation of the dual variables.

The gap between node i 's estimation of the average gradient and the true average gradient is further bounded as

$$\|\bar{\mathbf{z}}^t - \mathbf{z}_i^t\|_* \leq \sum_{s=1}^{t-1} L \|A^{t-s} \mathbf{e}_i - \mathbf{1}_n/n\|_1 + 2L, \quad (3.9)$$

which relies on the underlying combination matrix A . For edge sampling algorithms, the standard communication matrix A is replaced by a stochastic matrix B ¹, and we claim the following lemma:

¹In the following proof, we let $B = B(t)$ for simplicity of notations.

Lemma 3.4. *For edge sampling algorithm, the following probability holds:*

$$\mathbb{P} \left[\|B^{t-s} \mathbf{e}_i - \mathbf{1}_n/n\|_2 \geq \frac{1}{T^2 n} \right] \leq T^4 n^2 [\kappa(z)]^{t-s} \quad (3.10)$$

Proof: Consider the sequence $\mathbf{v}^t = B^t \mathbf{e}_i - \mathbf{1}_n/n$, and we have

$$\mathbf{v}^{t+1} = B^{t+1} \mathbf{e}_i - \mathbf{1}_n/n = B(B^t \mathbf{e}_i - \mathbf{1}_n/n) = B\mathbf{v}^t$$

Therefore, we can calculate the second moment of $\mathbf{v}^t$ as

$$\mathbb{E} [\langle \mathbf{v}^{t+1}, \mathbf{v}^{t+1} \rangle \mid \mathbf{v}^t] = \mathbb{E} [(B\mathbf{v}^t)^\top (B\mathbf{v}^t) \mid \mathbf{v}^t] = (\mathbf{v}^t)^\top \mathbb{E} [B^\top B] \mathbf{v}^t = (\mathbf{v}^t)^\top \mathbb{E} [B^2] \mathbf{v}^t. \quad (3.11)$$

Since B^2 is guaranteed to be doubly stochastic, we conclude that $\mathbf{1}_n/n$ is the first eigenvector of $\mathbb{E}[B^2]$ corresponding to the largest eigenvalue 1. To proceed, we notice that $\mathbf{v}^t$ is orthogonal to $\mathbf{1}_n/n$ and use Rayleigh quotient to derive

$$(\mathbf{v}^t)^\top \mathbb{E} [B^2] \mathbf{v}^t \leq \|\mathbf{v}^t\|_2^2 \lambda_2 (\mathbb{E} [B^2]). \quad (3.12)$$

Telescoping (3.11)(3.12) and applying Markov's inequality, we have

$$\begin{aligned} \mathbb{P} \left[\|\mathbf{v}^t\|_2 \geq \frac{1}{T^2 n} \right] &= \mathbb{P} \left[\|\mathbf{v}^t\|_2^2 \geq \frac{1}{T^4 n^2} \right] \leq T^4 n^2 \mathbb{E} [\|\mathbf{v}^t\|_2^2] \\ &\leq T^4 n^2 \lambda_2 (\mathbb{E} [B^2])^t \|\mathbf{v}^0\|_2^2 \leq T^4 n^2 \lambda_2 (\mathbb{E} [B^2])^t = T^4 n^2 [\kappa(z)]^t. \end{aligned}$$

Replace t with $t - s$ leads to (3.10), and this concludes the proof. $\square$

With the above lemma, we can now proceed the proof for Theorem 3.1. Similar to [37], we consider the case $s \leq T - \frac{3 \log(T^2 n)}{\log((\kappa(z))^{-1})} := \hat{t}$. According to (3.10) we immediately have

$$\mathbb{P} \left[\|B^{T-s} \mathbf{e}_i - \mathbf{1}_n/n\|_2 \geq \frac{1}{T^2 n} \right] \leq \frac{1}{T^2 n}.$$

Now consider Eq (3.9) and replace the deterministic communication matrix A with our stochastic matrix B . Notice $\sum_{s=1}^{t-1} L \|B^{t-s} \mathbf{e}_i - \mathbf{1}_n/n\|_1 \leq$

$\sum_{s=1}^{T-1} L \|B^{T-s} \mathbf{e}_i - \mathbf{1}_n/n\|_1$ holds for all $t = 1, 2, \dots, T-1$, and we have

$$\begin{aligned}
\|\bar{\mathbf{z}}^t - \mathbf{z}_i^t\|_* &\leq \sum_{s=1}^{t-1} L \|B^{t-s} \mathbf{e}_i - \mathbf{1}_n/n\|_1 + 2L \\
&\leq \sum_{s=1}^{T-1} L \|B^{T-s} \mathbf{e}_i - \mathbf{1}_n/n\|_1 + 2L \\
&= \sum_{s=1}^{\hat{t}} L \|B^{T-s} \mathbf{e}_i - \mathbf{1}_n/n\|_1 + \sum_{\hat{t}+1}^{T-1} L \|B^{T-s} \mathbf{e}_i - \mathbf{1}_n/n\|_1 + 2L \\
&\leq \sum_{s=1}^{\hat{t}} L \sqrt{n} \|B^{T-s} \mathbf{e}_i - \mathbf{1}_n/n\|_2 + \sum_{\hat{t}+1}^{T-1} L \|B^{T-s} \mathbf{e}_i - \mathbf{1}_n/n\|_1 + 2L \\
&\leq L \sqrt{n} \frac{\hat{t}-1}{T^2 n} + 2L(T - \hat{t} - 2) + 2L \\
&\leq \frac{L}{T \sqrt{n}} + 6L \frac{\log(T^2 n)}{\log((\kappa(z))^{-1})} \\
&\leq \frac{L}{T \sqrt{n}} + 12L \frac{\log(T \sqrt{n})}{1 - \kappa(z)},
\end{aligned}$$

holds for all $t = [1, \dots, T-1]$ and $i \in [1, \dots, n]$ with probability at least $1 - \frac{1}{T^2 n}$, where in the last inequality we use $\log x \leq x - 1$.

Replacing $\|\bar{\mathbf{z}}^t - \mathbf{z}_i^t\|_*$ in Eq 3.9 with above upper bound, we have

$$f(\hat{\mathbf{w}}_i^T) - f(\mathbf{w}^*) \leq \frac{\psi(\mathbf{w}^*)}{T \eta^T} + \frac{L^2}{2T} \sum_{t=1}^T \eta^{t-1} + \frac{3L^2}{T} \left(\frac{12 \log(T \sqrt{n})}{1 - \kappa(z)} + \frac{1}{T \sqrt{n}} \right) \sum_{t=1}^T \eta^t \quad (3.13)$$

holds with probability at least $1 - \frac{1}{T^2 n}$.

setting the step size $\eta^t \propto \frac{R \sqrt{1 - \kappa(z)}}{L \sqrt{t}}$, we have

$$f(\hat{\mathbf{w}}_i^T) - f(\mathbf{w}^*) \leq c \frac{RL}{\sqrt{T}} \frac{\log(T \sqrt{n})}{\sqrt{1 - \kappa(z)}}$$

holds with probability at least $1 - \frac{1}{T^2 n}$ with a universal constant c . This concludes the proof. $\square$

Chapter 4

Online Federated Convex-Concave Optimization

In the previous chapter, we have studied optimization of learning convex problems on graphs with communication-efficient strategies, where we assume the local problems are unconstrained. In this chapter, we extend the problem to the convex-concave cases, where there may exist local or global constraints and each local agents may consider solving the convex-concave cases in the adaptation phase.

Local constraints are quite common in optimization problems, such as budget limitation in portfolio selection, prior information adaptation in data fitting and projection into feasible areas in machine learning [32, 51]. On decentralized networks where communication is limited to each node's direct neighbors, one practical solution for such a constrained problem would be constructing a local Lagrangian formulation and solving the convex-concave problem by iterative updating, and then sharing local weights via communication [59].

Apart from the constrained optimization, many problems are naturally cast as convex-concave problems, such as min-max theorem in game theory [60], primal-dual methods in optimization theory [61] and online AUC maximization in machine learning [62]. Whether we can distribute these problems over decentralized networks via online learning and how the networks affect their convergence rates still remain unknown.

Motivated by these challenges, in this chapter, we focus on the developing and analyzing algorithms in seeking the optimal saddle point for online federated convex-concave learning over decentralized networks. We extend the ‘combine-and-then-adapt’ (CTA) strategy to this convex-concave setting to achieve the saddle point by iteratively using neighbor communication and local computation. With clear $\mathcal{O}(\sqrt{T})$ upper and lower regret bounds, we theoretically guarantee the performance of our proposed algorithm, as well as revealing its relationship with the network size and topology. To the best of our knowledge, this is the first established theoretical regret analysis for decentralized online convex-concave problems. This theoretical result is further confirmed by real-world experiments for online AUC maximization, which turns out to be a byproduct of our general solution but stands out as the first distributed online AUC maximization algorithm.

4.1 Decentralized Online Saddle Point Algorithm

In this section, we provide a formal formulation of our proposed online federated convex-concave learning (denoted as ‘D-OSP’) to reach the saddle point on decentralized networks.

4.1.1 Preliminaries

Learning on graphs: Let $G = (V, E)$ denotes an undirected graph with vertex set $V = \{1, \dots, m\}$ and edge set $E \subset V \times V$. For convex-concave problems, each learning node $i \in V$ is required to generate a learning sequence $(\mathbf{w}_i^t, \mathbf{d}_i^t)$ from a convex compact set $(\mathcal{W}, \mathcal{D})$, with $\mathbf{w}_i^t$ denoting the convex part and $\mathbf{d}_i^t$ denoting the concave part. Then the adversary validates each learner’s decision with a convex-concave loss function $f(\mathbf{w}, \mathbf{d})$ and each learner i updates its learning weight based on the loss. The communication is limited by the graph G : learner i can only communicate with its immediate neighbors $N(i) = \{j \in V \mid (i, j) \in E\}$. This is possible by setting $a_{i,j} = 0$ if $(i, j) \notin E$. For simplicity of analysis, we assume the combination matrix A is doubly stochastic, namely $A^\top \mathbf{1} = \mathbf{1}$ and $A\mathbf{1} = \mathbf{1}$.

Saddle-point problem: For a convex-concave function $f(\mathbf{w}, \mathbf{d})$, the target is approaching the saddle point $(\mathbf{w}^*, \mathbf{d}^*)$:

$$\mathbf{w}^*, \mathbf{d}^* = \inf_{\mathbf{w} \in \Omega} \sup_{\mathbf{d} \in \mathcal{D}} f(\mathbf{w}, \mathbf{d}).$$

It's straightforward to know the strong max-min property of saddle point problem:

$$f(\mathbf{w}^*, \mathbf{d}) \leq f(\mathbf{w}^*, \mathbf{d}^*) \leq f(\mathbf{w}, \mathbf{d}^*).$$

To see the necessities of finding this saddle point of convex-concave problems, we consider a generic constrained convex optimization problem,

$$\begin{aligned} \min_{\mathbf{w} \in \Omega} \quad & p(\mathbf{w}) \\ \text{s.t.} \quad & g(\mathbf{w}) \leq 0, h(\mathbf{w}) = 0. \end{aligned}$$

The common approach is forming a Lagrangian problem as $f(\mathbf{w}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = p(\mathbf{w}) + \boldsymbol{\mu}^\top g(\mathbf{w}) + \boldsymbol{\lambda}^\top h(\mathbf{w})$ and seeking for its saddle point $(\mathbf{w}^*, \boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)$ [63], where $\mathbf{w}^*$ minimizes the original problem and $(\boldsymbol{\mu}^*, \boldsymbol{\lambda}^*)$ maximize the dual problem.

4.1.2 Problem Formulation

Learning Goal: In this thesis, we consider a general convex-concave function $f(\mathbf{w}, \mathbf{d})$, which is convex for the first vector $\mathbf{w}$ and concave for the second vector $\mathbf{d}$. We adopt the standard approach [13] of measuring the performance of online algorithms by calculating the accumulative gap between our sequential predictions and the optimal solution:

$$R_T := \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{w}_i^t, \mathbf{d}_i^t) - f(\mathbf{w}^*, \mathbf{d}^*)], \quad (4.1)$$

which is also known as the ‘Regret’ of online learning.

However, it's worth mentioning that there exist some differences between the traditional regret defined for online convex optimization and our target listed above. For convex optimization, the regret is always non-negative, while the regret R_T in Eq (4.1)

can be negative due to the concave part involved. Therefore, the goal of online convex-concave optimization is minimizing the absolute value of the regret $|R_T|$ rather than the regret itself¹. We shall establish both upper and lower bounds for R_T to guarantee the convergence of our proposed algorithm in Part III.

Learning Structure: We consider the learning structure named as ‘Combine-and-then-adapt’ (CTA) method (Chp 7, [Sayed et al.](#)). For node i , it first combines the learning weight from its immediate neighbors based on a fixed combination matrix A :

$$\mathbf{v}_i^t = \sum_{j=1}^m a_{i,j} \mathbf{w}_j^t; \quad (4.2)$$

$$\mathbf{c}_i^t = \sum_{j=1}^m a_{i,j} \mathbf{d}_j^t. \quad (4.3)$$

Then it receives an instance and makes a prediction based on $(\mathbf{v}_i^t, \mathbf{c}_i^t)$. Upon receiving the loss $f(\mathbf{v}_i^t, \mathbf{c}_i^t)$, it updates its learning weight through online gradient descent(ascend):

$$\mathbf{w}_i^{t+1} = \mathbf{v}_i^t - \eta_1 \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{v}_i^t}; \quad (4.4)$$

$$\mathbf{d}_i^{t+1} = \mathbf{c}_i^t + \eta_2 \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{c}_i^t}. \quad (4.5)$$

For the convenience of notations, we denote the information flow in Eq (4.4) as

$$\begin{aligned} \mathbf{p}_i^{t+1} &= \mathbf{w}_i^{t+1} - \mathbf{v}_i^t = -\eta_1 \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{v}_i^t}; \\ \mathbf{q}_i^{t+1} &= \mathbf{d}_i^{t+1} - \mathbf{c}_i^t = \eta_2 \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{c}_i^t}, \end{aligned}$$

and we make a smooth assumption that there exist L_w, L_d to bound the partial derivatives, namely

$$\left\| \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{v}_i^t} \right\| \leq L_w;$$

¹Considering a convex-concave function $f(\mathbf{w}, \mathbf{d}) = \mathbf{w}^2 - \mathbf{d}^2$, our target is minimizing the gap $R_T = \sum [f(\mathbf{w}, \mathbf{d}) - f(\mathbf{0}, \mathbf{0})]$. By fixing $\mathbf{w} = \mathbf{0}$, the regret is always non-positive while fixing $\mathbf{d} = \mathbf{0}$ leads to a non-negative regret. This gives a simple illustration why we need to give both upper and lower bounds of R_T in the following part.

$$\left\| \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{c}_i^t} \right\| \leq L_d, \quad (4.6)$$

are true for all $i \in [1, \dots, m]$ and $t \in [1, \dots, T]$.

4.1.2.1 Auxiliary Sequences

We define two auxiliary sequences to measure the average weight on t -th round:

$$\bar{\mathbf{w}}^t = \frac{1}{m} \sum_{i=1}^m \mathbf{w}_i^t, \quad \bar{\mathbf{d}}^t = \frac{1}{m} \sum_{i=1}^m \mathbf{d}_i^t, \quad (4.7)$$

Their updating rules can be derived as

$$\begin{aligned} \bar{\mathbf{w}}^{t+1} &\stackrel{(4.7)(4.4)}{=} \frac{1}{m} \sum_{i=1}^m (\mathbf{v}_i^t + \mathbf{p}_i^{t+1}) \\ &\stackrel{(4.2)}{=} \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^m a_{i,j} \mathbf{w}_j^t + \frac{1}{m} \sum_{i=1}^m \mathbf{p}_i^{t+1} \\ &= \frac{1}{m} \sum_{j=1}^m \mathbf{w}_j^t + \frac{1}{m} \sum_{i=1}^m \mathbf{p}_i^{t+1} \\ &= \bar{\mathbf{w}}^t + \frac{1}{m} \sum_{i=1}^m \mathbf{p}_i^{t+1}. \end{aligned} \quad (4.8)$$

In the above equation, we use the property of stochastic matrix $\mathbf{A}$: $\sum_{i=1}^m a_{i,j} = 1$.

Similarly, we have

$$\bar{\mathbf{d}}^{t+1} = \bar{\mathbf{d}}^t + \frac{1}{m} \sum_{i=1}^m \mathbf{q}_i^{t+1}. \quad (4.9)$$

These two equations indicate that average weight $(\bar{\mathbf{w}}, \bar{\mathbf{d}})$ of the network almost evolves like the standard online gradient methods, except for the difference that each node calculates its partial derivatives $\mathbf{p}_i^{t+1}, \mathbf{q}_i^{t+1}$ with local copy $\mathbf{v}_i^t, \mathbf{c}_i^t$ instead of the average value. This gives the intuition that the whole network will converge to the optimal $\mathbf{w}^*$ as long as we can control the weight difference, which we will bound in Lemma 1 in the next section.

Algorithm 3 Decentralized Online Saddle Point Algorithm (D-OSP)

```

1: Initialize: Combination matrix  $A$ , step-size  $\eta_1$  and  $\eta_2$ 
2: Initialize:  $\mathbf{w}_i^0 = \mathbf{0}, \mathbf{d}_i^0 = \mathbf{0}$  for  $\forall i \in [1, \dots, m]$ 
3: for  $t = 1, \dots, T$  do
4:   for Each learner  $i \in V$  do
5:     Combine weights from its direct neighbor:
6:     Convex variable:  $\mathbf{v}_i^t = \sum_{j=1}^m a_{i,j} \mathbf{w}_j^t$ 
7:     Concave variable:  $\mathbf{c}_i^t = \sum_{j=1}^m a_{i,j} \mathbf{d}_j^t$ 
8:     Receive an instance  $(\mathbf{x}_i^t, y_i^t)$ 
9:     Make a prediction based on  $(\mathbf{v}_i^t, \mathbf{c}_i^t)$ 
10:    Compute the loss function  $f(\mathbf{v}_i^t, \mathbf{c}_i^t)$ 
11:    Adapt to the loss:
12:    Convex variable:  $\mathbf{w}_i^{t+1} = \mathbf{v}_i^t - \eta_1 \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{v}_i^t}$ 
13:    Concave variable:  $\mathbf{d}_i^{t+1} = \mathbf{c}_i^t + \eta_2 \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{c}_i^t}$ 
14:   end for
15: end for
16: Output:  $\mathbf{w}_i^T$  and  $\mathbf{d}_i^T$  for  $i = 1, \dots, m$ 

```

Equipped with the learning strategy (Eqs (4.2)(4.4)) and these auxiliary sequences, we now summarize the decentralized online saddle point algorithm in Algorithm 3.

4.2 Theoretical Analysis

In this section, we provide detailed regret analysis for our proposed algorithm with both upper and lower bounds.

4.2.1 Network Error Bound

Without loss of generality, let's assume $\mathbf{w}_i^0 = \mathbf{0}$ and $\mathbf{d}_i^0 = \mathbf{0}$. By defining a chain matrix $\Phi(k, s) = A^{k-s}$ ($k \geq s \geq 0$), the evolution of the sequential weight on j -th node can be derived as

$$\begin{aligned}
 \mathbf{w}_j^{t+1} &= \mathbf{p}_j^{t+1} + \sum_{l=1}^t \sum_{i=1}^m [\Phi(t+1, l)]_{j,i} \mathbf{p}_i^l \\
 \mathbf{d}_j^{t+1} &= \mathbf{q}_j^{t+1} + \sum_{l=1}^t \sum_{i=1}^m [\Phi(t+1, l)]_{j,i} \mathbf{q}_i^l
 \end{aligned} \tag{4.10}$$

The *network error* is defined as the gap between the weight on each node $(\mathbf{w}_j^t, \mathbf{d}_j^t)$ and the average value $(\bar{\mathbf{w}}^t, \bar{\mathbf{d}}^t)$ across the network, and it plays a key role in the theoretical analysis for distributed online learning [37]. Therefore, we shall first bound this term before proceeding the final regret analysis.

Lemma 4.1. Denote $\sigma_2(A)$ as the second-largest singular value of matrix A and $1 - \sigma_2(A)$ as the corresponding spectral gap, the network error term can be bounded as:

$$\|\bar{\mathbf{w}}^{t+1} - \mathbf{w}_j^{t+1}\| \leq \eta_1 L_w \left(\frac{\sqrt{m}\sigma_2(A)}{1 - \sigma_2(A)} + 2 \right); \quad (4.11)$$

$$\|\bar{\mathbf{d}}^{t+1} - \mathbf{d}_j^{t+1}\| \leq \eta_2 L_d \left(\frac{\sqrt{m}\sigma_2(A)}{1 - \sigma_2(A)} + 2 \right). \quad (4.12)$$

4.2.2 Upper Bound

To derive an upper bound for our regret¹, we rewrite Eq (4.1) as

$$\begin{aligned} R_T &= \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\ &= \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t)] \\ &\quad + \sum_{t=1}^T \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\ &\leq \sum_{t=1}^T \sum_{i=1}^m L_w \|\bar{\mathbf{w}}^t - \mathbf{v}_i^t\| + \sum_{t=1}^T \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \end{aligned} \quad (4.13)$$

The last inequality holds due to the fact

$$f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) \leq \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{v}_i^t} (\mathbf{v}_i^t - \bar{\mathbf{w}}^t) \leq L_w \|\mathbf{v}_i^t - \bar{\mathbf{w}}^t\|.$$

The following lemma bounds the last term in (4.13).

¹Here we use the prediction discrepancy based on $(\mathbf{v}_i^t, \mathbf{c}_i^t)$ rather than $(\mathbf{w}_i^t, \mathbf{d}_i^t)$ since the algorithm uses $(\mathbf{v}_i^t, \mathbf{c}_i^t)$ to do the prediction work, and $(\mathbf{w}^*, \mathbf{d}^*) = (\mathbf{v}^*, \mathbf{c}^*)$ since $\mathbf{w}^* = \sum_{i=1}^m a_{i,j} \mathbf{v}^* = \mathbf{v}^*$ and $\mathbf{d}^* = \sum_{i=1}^m a_{i,j} \mathbf{c}^* = \mathbf{c}^*$.

Lemma 4.2. *The difference between $f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t)$ and $f(\mathbf{v}^*, \mathbf{c}^*)$ can be bounded as:*

$$\begin{aligned} & \sum_{t=1}^T \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\ & \leq \frac{1}{2\eta_1} \sum_{i=1}^m \|\mathbf{v}_i^0 - \mathbf{v}^*\|_2^2 + L_w \sum_{t=1}^T \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{mT}{2} \eta_1 L_w^2. \end{aligned}$$

Equipped with these two lemmas, we are able to derive an upper bound for our learning regret in the following theorem.

Theorem 4.3. *Assuming $\mathbf{v}_i^0 = 0$ is true for all learning nodes, the D-OSP algorithm attains the following regret upper bound:*

$$R_T \leq \frac{m}{2\eta_1} \|\mathbf{v}^*\|_2^2 + \eta_1 \left(\frac{2\sqrt{m}\sigma_2(A)}{1 - \sigma_2(A)} + \frac{9}{2} \right) mT L_w^2.$$

Proof: The proof is straightforward by considering Eq (4.13) and Lemma 4.1, 4.2:

$$\begin{aligned} R_T & \stackrel{(4.13)}{\leq} \sum_{t=1}^T \sum_{i=1}^m L_w \|\bar{\mathbf{w}}^t - \mathbf{v}_i^t\| \\ & + \sum_{t=1}^T \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\ & \stackrel{\text{Lemma 2}}{\leq} \sum_{t=1}^T \sum_{i=1}^m L_w \|\bar{\mathbf{w}}^t - \mathbf{v}_i^t\| + \frac{1}{2\eta_1} \sum_{i=1}^m \|\mathbf{v}_i^0 - \mathbf{v}^*\|_2^2 \\ & + L_w \sum_{t=1}^T \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{mT}{2} \eta_1 L_w^2 \\ & \stackrel{(4.2)}{=} \sum_{t=1}^T \sum_{i=1}^m L_w \|\bar{\mathbf{w}}^t - \sum_{j=1}^m a_{i,j} \mathbf{w}_j^t\| + \frac{1}{2\eta_1} \sum_{i=1}^m \|\mathbf{v}_i^0 - \mathbf{v}^*\|_2^2 \\ & + L_w \sum_{t=1}^T \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{mT}{2} \eta_1 L_w^2 \\ & \leq \sum_{t=1}^T \sum_{i=1}^m \sum_{j=1}^m a_{i,j} L_w \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{1}{2\eta_1} \sum_{i=1}^m \|\mathbf{v}_i^0 - \mathbf{v}^*\|_2^2 \\ & + L_w \sum_{t=1}^T \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{mT}{2} \eta_1 L_w^2 \end{aligned}$$

$$\begin{aligned}
&\leq \sum_{t=1}^T \sum_{j=1}^m L_w \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{1}{2\eta_1} \sum_{i=1}^m \|\mathbf{v}_i^0 - \mathbf{v}^*\|_2^2 \\
&\quad + L_w \sum_{t=1}^T \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{mT}{2} \eta_1 L_w^2 \\
&\stackrel{(4.11)}{\leq} 2\eta_1 L_w^2 mT \left(\frac{\sqrt{m}\sigma_2(A)}{1 - \sigma_2(A)} + 2 \right) \\
&\quad + \frac{m}{2\eta_1} \|\mathbf{v}^*\|_2^2 + \frac{mT}{2} \eta_1 L_w^2 \\
&= \frac{m}{2\eta_1} \|\mathbf{v}^*\|_2^2 + \eta_1 \left(\frac{2\sqrt{m}\sigma_2(A)}{1 - \sigma_2(A)} + \frac{9}{2} \right) mT L_w^2
\end{aligned}$$

This concludes the proof. $\square$

Corollary 4.4. *The proposed D-OSP algorithm achieves an $\mathcal{O}(\sqrt{T})$ regret upper bound by setting*

$$\eta_1 = \sqrt{\frac{\|\mathbf{v}^*\|_2^2}{2T\Delta}} \frac{1}{L_w},$$

where $\Delta = \frac{2\sqrt{m}\sigma_2(A)}{1 - \sigma_2(A)} + \frac{9}{2}$.

Remark: (1) For typical online convex learning, $\mathcal{O}(\sqrt{T})$ has been proved to be the optimal regret for the general case [13]. For online convex-concave learning, this regret upper bound is also in the similar order to many centralized counterparts (e.g., $\mathcal{O}(\sqrt{T})$ regret bound in [62, 64]).

(2) Our theorem and its corollary provide the first numerical estimation of how the network topology and scale affect the final performance of distributed online convex-concave learning. The intuition that a well-connected network is expected to perform better than those poor-connected networks can be easily obtained since its second largest eigenvalue σ_2 is smaller and leads to a lower regret. In the meanwhile, communication in graphs with smaller node numbers m is apparently more efficient and therefore resulting in lower regrets.

4.2.3 Lower Bound

To derive a lower bound for our regret, we rewrite Eq (4.1) as

$$\begin{aligned}
R_T &= \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\
&= \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t)] \\
&\quad + \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\
&\geq \sum_{t=1}^T \sum_{i=1}^m -L_d \|\bar{\mathbf{d}}^t - \mathbf{c}_i^t\| + \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \tag{4.14}
\end{aligned}$$

In the following lemma, we shall derive a lower bound for the last term.

Lemma 4.5. *The difference between $f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t)$ and $f(\mathbf{v}^*, \mathbf{c}^*)$ can be bounded as:*

$$\begin{aligned}
&\sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\
&\geq -\frac{1}{2\eta_2} \sum_{i=1}^m \|\mathbf{c}_i^0 - \mathbf{c}^*\|_2^2 - L_d \sum_{t=1}^T \sum_{i=1}^m \|\bar{\mathbf{d}}^t - \mathbf{d}_j^t\| - \frac{\eta_2}{2} m T L_d^2
\end{aligned}$$

Equipped with Lemma 4.1 and Lemma 4.5, we are able to derive a lower bound for R_T in the following theorem.

Theorem 4.6. *Assuming $\mathbf{c}_i^0$ is true for all learning nodes, the D-OSP algorithm attains the following regret lower bound:*

$$R_T \geq -\frac{m}{2\eta_2} \|\mathbf{c}^*\|_2^2 - \eta_2 \left(\frac{2\sqrt{m}\sigma_2(A)}{1 - \sigma_2(A)} + \frac{9}{2} \right) m T L_d^2.$$

Corollary 4.7. *The proposed D-OSP algorithm achieves an $\mathcal{O}(\sqrt{T})$ regret lower bound by setting*

$$\eta_2 = \sqrt{\frac{\|\mathbf{c}^*\|^2}{2T\Delta}} \frac{1}{L_d}$$

where $\Delta = \frac{2\sqrt{m}\sigma_2(A)}{1 - \sigma_2(A)} + \frac{9}{2}$.

Remark: The lower bound is also related to the topology and size of the network. According to the theorem and its corollary, a well-connected network is expected to perform better than those poor-connected counterparts, while a smaller network tends to outperform large-scale competitors. We shall examine these properties in the following part.

4.3 Experiments

To evaluate the performance of our proposed D-OSP algorithm, we conduct a series of experiments for distributed online convex-concave problems in this part. For source code and implementation details, see Supplementary Material E.

4.3.1 Distributed Online AUC Maximization

Area under ROC curve (AUC) is a popular metric [62, 64–67] which quantifies their ability to correctly rank any positive instance with regards to a randomly chosen negative instance. Previous research [62] has proved that if the square loss is adopted, it is equivalent to be a convex-concave optimization problem,

$$\arg \min_{\substack{\hat{\mathbf{w}} \in \mathcal{W} \\ (\beta, \gamma) \in \mathbb{R}^2}} \arg \max_{\alpha \in \mathbb{R}} f(\hat{\mathbf{w}}, \beta, \gamma, \alpha) := \int_{\mathcal{Z}} F(\hat{\mathbf{w}}, \beta, \mathbf{g}, \alpha; z) d\rho(z), \quad (4.15)$$

where

$$\begin{aligned} F(\hat{\mathbf{w}}, \beta, \mathbf{g}, \alpha; z) &= (1 - p)(\hat{\mathbf{w}}^\top \mathbf{x} - \beta)^2 \mathbb{I}_{[y=1]} \\ &+ p(\hat{\mathbf{w}}^\top \mathbf{x} - \mathbf{g})^2 \mathbb{I}_{[y=-1]} - p(1 - p)\alpha^2 \\ &+ 2(1 + \alpha) \left(p\hat{\mathbf{w}}^\top \mathbf{x} \mathbb{I}_{[y=-1]} - (1 - p)\hat{\mathbf{w}}^\top \mathbf{x} \mathbb{I}_{[y=1]} \right). \end{aligned}$$

Here $p = \Pr(y = 1)$ denotes the possibility of positive labels and $\beta, \mathbf{g}, \alpha$ are three auxiliary variables. Proof of this equivalence is provided in the appendix.

In the following part, we focus on distributing this convex-concave problem on decentralized networks to validate the efficiency of our proposed algorithm, which is also

the first distributed online AUC maximization algorithm. To unify the notations, we compress the variables in groups, namely $\mathbf{w} = (\hat{\mathbf{w}}, \beta, \gamma)$ and $\mathbf{d} = \alpha$.

4.3.2 Algorithms

We consider the AUC maximization algorithm as a convex-concave problem, and the corresponding algorithm is listed in Alg 4.

Algorithm 4 Decentralized Distributed Online AUC Maximization

- 1: **Initialize:** Combination matrix A , step-size η_1 and η_2
 - 2: **Initialize:** $\mathbf{w}_i^0 = \mathbf{0}, \mathbf{d}_i^0 = 0$ for $\forall i \in [1, \dots, m]$
 - 3: **for** $t = 1, \dots, T$ **do**
 - 4: **for** Each learner $i \in V$ **do**
 - 5: Combine weights from its direct neighbor:
 - 6: Convex: $\mathbf{v}_i^t = \sum_{j=1}^m a_{i,j} \mathbf{w}_j^t$
 - 7: Concave: $\mathbf{c}_i^t = \sum_{j=1}^m a_{i,j} \mathbf{d}_j^t$
 - 8: Receive an instance $(\mathbf{x}_i^t, y_i^t)$ and predict AUC based on $(\mathbf{v}_i^t, \mathbf{d}_i^t)$
 - 9: Compute the partial gradient $\partial_{\mathbf{v}} F$ and $\partial_{\mathbf{c}} F$
 - 10: Adapt to the loss:
 - 11: Convex: $\mathbf{w}_i^{t+1} = P_{\Omega_1}(\mathbf{v}_i^t - \eta_1 \partial_{\mathbf{v}} F(\mathbf{v}_i^t, \mathbf{c}_i^t))$
 - 12: Concave: $\mathbf{d}_i^{t+1} = P_{\Omega_2}(\mathbf{c}_i^t + \eta_2 \partial_{\mathbf{c}} F(\mathbf{v}_i^t, \mathbf{c}_i^t))$
 - 13: **end for**
 - 14: **end for**
 - 15: **Output:** $\mathbf{w}_i^T$ for $i = 1, \dots, m$
-

4.3.3 Datasets and Networks

We perform our D-OSP algorithm on 6 datasets^{1 2} that have been used in previous centralized AUC studies [62, 64], covering both balanced data (e.g. *covtype* dataset) and imbalanced data (e.g. *w8a* dataset).

¹<https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/>

²<http://archive.ics.uci.edu/ml/index.php>

Datasets	# Instances	# Dimensions	T_-/T_+
<i>magic04</i>	19020	10	0.542
<i>covtype</i>	581012	54	0.952
<i>a9a</i>	48842	123	3.179
<i>skin</i>	245057	3	3.818
<i>ijcnn1</i>	141691	22	9.445
<i>w8a</i>	64700	300	32.471

TABLE 4.1: Datasets Information

To investigate the influence of different network topologies, we conduct our experiments on three types of graphs with different levels of connectivity.

- *Complete graph*. All nodes are connected to each other, and this represents the highest level of connectivity.
- *Watts-Strogatz graph*. A random graph is generated with two tunable parameters: the average degree k and the rewiring probability p [68]. We set $k = 4$ and $p = 0.3$ in our experiments.
- *Cycle graph*. All nodes form a circle and each node is only connected to its three neighbors (including itself as a neighbor). This represents the lowest level of connectivity in our experiments.

To investigate the effect of network scale, we construct different networks with $m = 10, 50, 100, 500$ nodes and consider *centralized* online AUC maximization [62] as our benchmark.

4.3.4 Experimental Results

We report experimental results on the network with different node numbers and graph type in this section.

Table 4.2 shows the final AUC score for all datasets with different graph types and node numbers. We can easily observe the following facts: 1) the AUC score monotonously drops as the number of learning nodes increases; 2) Network with good connectivity (e.g., complete graph) almost consistently outperforms those with

Data	# Nodes	Complete	Watts-Strogatz	Cycle
<i>a9a</i>	Centralized	89.59 ± 0.04		
	10	89.24 ± 0.03	89.20 ± 0.02	88.62 ± 0.10
	50	87.58 ± 0.04	87.50 ± 0.03	86.78 ± 0.20
	100	86.08 ± 0.04	85.98 ± 0.06	85.25 ± 0.20
	500	83.32 ± 0.11	82.32 ± 0.13	79.33 ± 0.19
<i>w8a</i>	Centralized	93.80 ± 0.06		
	10	92.98 ± 0.09	92.89 ± 0.17	91.84 ± 0.52
	50	89.62 ± 0.30	89.80 ± 0.38	87.16 ± 1.12
	100	88.25 ± 0.37	88.20 ± 0.21	83.53 ± 1.48
	500	83.91 ± 0.50	82.38 ± 0.78	76.75 ± 0.50
<i>covtype</i>	Centralized	88.79 ± 0.32		
	10	88.70 ± 0.37	88.44 ± 0.33	85.51 ± 0.33
	50	85.91 ± 0.42	85.19 ± 0.79	74.57 ± 2.58
	100	85.14 ± 0.86	84.28 ± 0.67	70.90 ± 2.32
	500	83.18 ± 1.22	76.75 ± 0.33	63.15 ± 0.43
<i>ijcnn1</i>	Centralized	90.23 ± 0.32		
	10	90.15 ± 0.34	90.11 ± 0.21	87.25 ± 0.69
	50	89.02 ± 0.21	88.96 ± 0.18	81.40 ± 1.16
	100	88.20 ± 0.31	88.14 ± 0.19	76.73 ± 2.37
	500	87.16 ± 0.27	84.00 ± 0.39	70.57 ± 0.27
<i>magic04</i>	Centralized	80.18 ± 0.18		
	10	79.93 ± 0.18	79.82 ± 0.11	78.86 ± 0.40
	50	78.88 ± 0.40	78.24 ± 0.36	73.52 ± 1.43
	100	78.07 ± 0.71	76.03 ± 0.52	68.71 ± 0.66
	500	72.68 ± 2.36	63.28 ± 0.38	56.87 ± 0.29
<i>skin</i>	Centralized	93.86 ± 0.06		
	10	93.28 ± 0.08	93.21 ± 0.08	93.22 ± 0.17
	50	91.39 ± 0.16	91.38 ± 0.16	90.93 ± 1.04
	100	90.86 ± 0.19	90.98 ± 0.13	90.46 ± 1.15
	500	84.85 ± 0.20	84.72 ± 0.24	83.62 ± 0.78

TABLE 4.2: Experimental results(%) with different network types and scales

Figure 4.1: Effect of node numbers for online learning process on *a9a* and *w8a* datasets.

poor connectivity (e.g. cycle graph). These observations are in accordance with our theoretical results in Theorem 4.3 and 4.6 where networks with larger spectral gap $1 - \sigma_2(A)$ and less node number m have smaller regrets. The good performance on well-connected graphs with small node numbers validates the possibility of distributing large datasets (e.g., *covtype* dataset) to multiple learning nodes over decentralized networks. On the other hand, the AUC score significantly drops down on graphs with a large number of learning nodes but poor connectivity. This can be easily explained by our theoretical analysis, where the connectivity factor $\sigma_2/(1 - \sigma_2)$ is magnified by the node number m .

We investigate how the graph types and node numbers affect the convergence rate by recording the online learning process in Figure 4.14.2. When fixing T and σ_2 , graphs with large m actually leads to a larger gap between accumulative loss and the optimal one, which is clearly validated by our real world experiments that graphs with large learning nodes converge relatively slower than their counterparts in the first two sub-figures. In the following two sub-figures, we study the influence of network topology by setting the node number to 50 on *a9a* and *w9a* datasets, and the result is also consistent with our previous theoretical analysis where graphs with poor connectivity has larger regret value $|R_T|$ and are expected to converge slower during the online learning process.

To summarize, the empirical behavior of real-world experiments is in excellent agreement with our theoretical analysis.

Figure 4.2: Effect of graph types for online learning process on *a9a* and *w8a* datasets.

4.4 Chapter Summary

In this chapter, we present a general solution, named as 'Decentralized Online Saddle Point Algorithm' (D-OSP), to distribute online convex-concave optimization problem over decentralized networks and seek for its saddle point by iterative local updating and mutual communication. The convergence rate of our proposed algorithm is guaranteed with complete regret analysis, where we establish $\mathcal{O}(\sqrt{T})$ upper and lower bounds for our proposed algorithm. Our analysis also numerically reveals how the network topology and size affect the performance of our proposed algorithm, which is further verified by real-world experiments for distributed online AUC maximization on multiple datasets.

4.5 Appendix: Proof for Lemmas

4.5.1 Jensen's Inequalities

Let $\{\alpha_i\}$ denote a collection of nonnegative real coefficients that add up to one $\sum_{i=1}^m \alpha_i = 1$, $0 \leq \alpha_i \leq 1$. Jensen's inequality holds for any real-valued convex function $f(\mathbf{w})$:

$$f\left(\sum_{i=1}^m \alpha_i \mathbf{w}_i\right) \leq \sum_{i=1}^m \alpha_i f(\mathbf{w}_i).$$

In the following proof, we shall frequently use this property for Euclidean norm:

$$\left\|\sum_{i=1}^m \alpha_i \mathbf{w}_i\right\| \leq \sum_{i=1}^m \alpha_i \|\mathbf{w}_i\|. \quad (4.16)$$

4.5.2 Proof for Lemma 4.1

Lemma 4.1. Denote $\sigma_2(A)$ as the second-largest singular value of matrix A and $1 - \sigma_2(A)$ as the corresponding spectral gap, the network error term can be bounded as:

$$\|\bar{\mathbf{w}}^{t+1} - \mathbf{w}_j^{t+1}\| \leq \eta_1 L_w \left(\frac{\sqrt{m}\sigma_2(A)}{1 - \sigma_2(A)} + 2 \right); \quad (4.11)$$

$$\|\bar{\mathbf{d}}^{t+1} - \mathbf{d}_j^{t+1}\| \leq \eta_2 L_d \left(\frac{\sqrt{m}\sigma_2(A)}{1 - \sigma_2(A)} + 2 \right). \quad (4.12)$$

Proof: Telescoping Eq (4.8) and using Eq (5.17), we have

$$\begin{aligned} & \|\bar{\mathbf{w}}^{t+1} - \mathbf{w}_j^{t+1}\| \\ &= \left\| \frac{1}{m} \sum_{l=1}^{t+1} \sum_{i=1}^m \mathbf{p}_i^l - \mathbf{p}_j^{t+1} - \sum_{l=1}^t \sum_{i=1}^m [\Phi(t+1, l)]_{j,i} \mathbf{p}_i^l \right\| \\ &= \left\| \sum_{l=1}^t \sum_{i=1}^m \left(\frac{1}{m} - [\Phi(t+1, l)]_{j,i} \right) \mathbf{p}_i^l + \left(\frac{1}{m} \sum_{i=1}^m \mathbf{p}_i^{t+1} - \mathbf{p}_j^{t+1} \right) \right\| \\ &\leq \sum_{l=1}^t \sum_{i=1}^m \left| \frac{1}{m} - [\Phi(t+1, l)]_{j,i} \right| \|\mathbf{p}_i^l\| + \frac{1}{m} \sum_{i=1}^m \|\mathbf{p}_i^{t+1} - \mathbf{p}_j^{t+1}\| \\ &\stackrel{(4.6)}{\leq} \eta_1 L_w \sum_{l=1}^t \sum_{i=1}^m \left| \frac{1}{m} - [\Phi(t+1, l)]_{j,i} \right| + 2\eta_1 L_w \end{aligned}$$

$$\begin{aligned}
&= \eta_1 L_w \sum_{l=1}^t \left\| \frac{\mathbf{1}}{m} - [\Phi(t+1, l)]_j \right\|_1 + 2\eta_1 L_w \\
&\leq \eta_1 L_w \sum_{l=1}^t \sqrt{m} \sigma_2(A)^{t+1-l} + 2\eta_1 L_w \\
&\leq \eta_1 L_w \frac{\sqrt{m} \sigma_2(A)}{1 - \sigma_2(A)} + 2\eta_1 L_w.
\end{aligned}$$

In the proof above, we use the property of a stochastic matrix from [37] and [69]:

$$\|A_i^t - \mathbf{1}/m\|_1 \leq \sigma_2(A)^t \sqrt{m}.$$

This concludes the proof. $\square$

4.5.3 Proof for Lemma 4.2

Lemma 4.2. *The difference between $f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t)$ and $f(\mathbf{v}^*, \mathbf{c}^*)$ can be bounded as:*

$$\begin{aligned}
&\sum_{t=1}^T \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\
&\leq \frac{1}{2\eta_1} \sum_{i=1}^m \|\mathbf{v}_i^0 - \mathbf{v}^*\|_2^2 + L_w \sum_{t=1}^T \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{mT}{2} \eta_1 L_w^2.
\end{aligned}$$

Proof: By the updating rule of $\mathbf{w}_i^{t+1}$ in Eq (4.4), we have

$$\begin{aligned}
&\|\mathbf{w}_i^{t+1} - \mathbf{v}^*\|_2^2 \\
&= \left\| \left[\mathbf{v}_i^t - \eta_1 \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{v}_i^t} \right] - \mathbf{v}^* \right\|_2^2 \\
&= \|\mathbf{v}_i^t - \mathbf{v}^*\|_2^2 - 2\eta_1 \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{v}_i^t} (\mathbf{v}_i^t - \mathbf{v}^*) + \eta_1^2 L_w^2 \\
&\leq \|\mathbf{v}_i^t - \mathbf{v}^*\|_2^2 - 2\eta_1 [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)] + \eta_1^2 L_w^2 \tag{4.17}
\end{aligned}$$

The last inequality is due to the convexity of $\mathbf{v}$: $f(\mathbf{v}^*, \mathbf{c}_i^t) - f(\mathbf{v}_i^t, \mathbf{c}_i^t) \geq \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{v}_i^t} (\mathbf{v}^* - \mathbf{v}_i^t)$.

In the meantime, we have

$$\begin{aligned}
& \sum_{i=1}^m \|\mathbf{v}_i^{t+1} - \mathbf{v}^*\|_2^2 \\
& \stackrel{(4.2)}{=} \sum_{i=1}^m \left\| \sum_{j=1}^m a_{i,j} \mathbf{w}_j^{t+1} - \mathbf{v}^* \right\|_2^2 \\
& \stackrel{(4.16)}{\leq} \sum_{i=1}^m \sum_{j=1}^m a_{i,j} \|\mathbf{w}_j^{t+1} - \mathbf{v}^*\|_2^2 \\
& = \sum_{j=1}^m \|\mathbf{w}_j^{t+1} - \mathbf{v}^*\|_2^2.
\end{aligned} \tag{4.18}$$

Summing up Eq (4.17) for all i and replacing the l.h.s with above inequality, we obtain

$$\begin{aligned}
& \sum_{i=1}^m \|\mathbf{v}_i^{t+1} - \mathbf{v}^*\|_2^2 \\
& \leq \sum_{i=1}^m \|\mathbf{v}_i^t - \mathbf{v}^*\|_2^2 - 2\eta_1 \sum_{i=1}^m [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)] + m\eta_1^2 L_w^2.
\end{aligned} \tag{4.19}$$

To proceed, we further bound the second term on the r.h.s in Eq (4.19) as:

$$\begin{aligned}
& \sum_{i=1}^m [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)] \\
& = \sum_{i=1}^m [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t)] + \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)] \\
& \geq - \sum_{i=1}^m \left\| \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{v}_i^t} \right\| \|\bar{\mathbf{w}}^t - \mathbf{v}_i^t\| + \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)] \\
& \geq -L_w \sum_{i=1}^m \|\bar{\mathbf{w}}^t - \sum_{j=1}^m a_{i,j} \mathbf{w}_j^t\| + \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)] \\
& \stackrel{(4.16)}{\geq} -L_w \sum_{i=1}^m \sum_{j=1}^m a_{i,j} \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)] \\
& \stackrel{(?)}{=} -L_w \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)]
\end{aligned}$$

Therefore, Eq (4.19) can be written as

$$\begin{aligned}
& \sum_{i=1}^m \|\mathbf{v}_i^{t+1} - \mathbf{v}^*\|_2^2 \\
& \leq \sum_{i=1}^m \|\mathbf{v}_i^t - \mathbf{v}^*\|_2^2 + 2\eta_1 L_w \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| \\
& \quad - 2\eta_1 \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)] + m\eta_1^2 L_w^2.
\end{aligned} \tag{4.20}$$

Rewrite it as

$$\begin{aligned}
& \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)] \\
& \leq \frac{1}{2\eta_1} \left[\sum_{i=1}^m \|\mathbf{v}_i^t - \mathbf{v}^*\|_2^2 - \sum_{i=1}^m \|\mathbf{v}_i^{t+1} - \mathbf{v}^*\|_2^2 \right] \\
& \quad + L_w \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{m}{2} \eta_1 L_w^2.
\end{aligned} \tag{4.21}$$

Summing it from $t = 1$ to $t = T$, we have

$$\begin{aligned}
& \sum_{t=1}^T \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}_i^t)] \\
& \leq \frac{1}{2\eta_1} \sum_{i=1}^m \|\mathbf{v}_i^0 - \mathbf{v}^*\|_2^2 + L_w \sum_{t=1}^T \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{mT}{2} \eta_1 L_w^2.
\end{aligned}$$

Since f is concave for any $\mathbf{c}$, we have $f(\mathbf{v}^*, \mathbf{c}_i^t) \leq f(\mathbf{v}^*, \mathbf{c}^*)$, and therefore

$$\begin{aligned}
& \sum_{t=1}^T \sum_{i=1}^m [f(\bar{\mathbf{w}}^t, \mathbf{c}_i^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\
& \leq \frac{1}{2\eta_1} \sum_{i=1}^m \|\mathbf{v}_i^0 - \mathbf{v}^*\|_2^2 + L_w \sum_{t=1}^T \sum_{j=1}^m \|\bar{\mathbf{w}}^t - \mathbf{w}_j^t\| + \frac{mT}{2} \eta_1 L_w^2.
\end{aligned}$$

This concludes the proof. $\square$

4.5.4 Proof for Lemma 4.5

Lemma 4.5. *The difference between $f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t)$ and $f(\mathbf{v}^*, \mathbf{c}^*)$ can be bounded as:*

$$\begin{aligned} & \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\ & \geq -\frac{1}{2\eta_2} \sum_{i=1}^m \|\mathbf{c}_i^0 - \mathbf{c}^*\|_2^2 - L_d \sum_{t=1}^T \sum_{i=1}^m \|\bar{\mathbf{d}}^t - \mathbf{d}_j^t\| - \frac{\eta_2}{2} m T L_d^2 \end{aligned}$$

Proof: By the updating rule of $\mathbf{d}_i^{t+1}$ in Eq (4.4), we have

$$\begin{aligned} & \|\mathbf{d}_i^{t+1} - \mathbf{c}^*\|_2^2 \\ & = \left\| \left[\mathbf{c}_i^t + \eta_2 \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{c}_i^t} \right] - \mathbf{c}^* \right\|_2^2 \\ & = \|\mathbf{c}_i^t - \mathbf{c}^*\|_2^2 + 2\eta_2 \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{c}_i^t} (\mathbf{c}_i^t - \mathbf{c}^*) + \eta_2^2 L_d^2 \\ & \leq \|\mathbf{c}_i^t - \mathbf{c}^*\|_2^2 + 2\eta_2 [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\mathbf{v}_i^t, \mathbf{c}^*)] + \eta_2^2 L_d^2 \end{aligned} \quad (4.22)$$

The last inequality is due to the concavity of $\mathbf{d}$: $f(\mathbf{v}_i^t, \mathbf{c}^*) - f(\mathbf{v}_i^t, \mathbf{c}_i^t) \leq \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{c}_i^t} (\mathbf{c}^* - \mathbf{c}_i^t)$.

Similar to (4.18), we can derive

$$\sum_{i=1}^m \|\mathbf{c}_i^{t+1} - \mathbf{c}^*\|_2^2 \leq \sum_{i=1}^m \|\mathbf{d}_i^{t+1} - \mathbf{c}^*\|_2^2 \quad (4.23)$$

and

$$\begin{aligned} & \sum_{i=1}^m [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\mathbf{v}_i^t, \mathbf{c}^*)] \\ & = \sum_{i=1}^m [f(\mathbf{v}_i^t, \mathbf{c}_i^t) - f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t)] + \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}_i^t, \mathbf{c}^*)] \\ & \leq \sum_{i=1}^m \left\| \frac{\partial f(\mathbf{v}_i^t, \mathbf{c}_i^t)}{\partial \mathbf{c}_i^t} \right\| \|\bar{\mathbf{d}}^t - \mathbf{c}_i^t\| + \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}_i^t, \mathbf{c}^*)] \\ & \leq L \sum_{i=1}^m \|\bar{\mathbf{d}}^t - \sum_{j=1}^m a_{i,j} \mathbf{d}_j^t\| + \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}_i^t, \mathbf{c}^*)] \end{aligned}$$

$$\begin{aligned}
&\leq L \sum_{i=1}^m \sum_{j=1}^m a_{i,j} \|\bar{\mathbf{d}}^t - \mathbf{d}_j^t\| + \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}_i^t, \mathbf{c}^*)] \\
&= L_d \sum_{i=1}^m \|\bar{\mathbf{d}}^t - \mathbf{d}_j^t\| + \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}_i^t, \mathbf{c}^*)]. \tag{4.24}
\end{aligned}$$

Summing (21) up from $i = 1$ to m and replacing the counterparts with Eq (4.23)(4.24), we obtain

$$\begin{aligned}
&\sum_{i=1}^m \|\mathbf{c}_i^{t+1} - \mathbf{c}^*\|_2^2 \\
&\leq \sum_{i=1}^m \|\mathbf{c}_i^t - \mathbf{c}^*\|_2^2 + 2\eta_2 L_d \sum_{i=1}^m \|\bar{\mathbf{d}}^t - \mathbf{d}_j^t\| \\
&\quad + 2\eta_2 \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}_i^t, \mathbf{c}^*)] + m\eta_2^2 L_d^2
\end{aligned}$$

Rewrite it as

$$\begin{aligned}
&\sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}_i^t, \mathbf{c}^*)] \\
&\geq \frac{1}{2\eta_2} \left[\sum_{i=1}^m \|\mathbf{c}_i^{t+1} - \mathbf{c}^*\|_2^2 - \sum_{i=1}^m \|\mathbf{c}_i^t - \mathbf{c}^*\|_2^2 \right] \\
&\quad - L_d \sum_{i=1}^m \|\bar{\mathbf{d}}^t - \mathbf{d}_j^t\| - \frac{\eta_2}{2} m L_d^2
\end{aligned}$$

Summing it up, we have

$$\begin{aligned}
&\sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}_i^t, \mathbf{c}^*)] \\
&\geq -\frac{1}{2\eta_2} \sum_{i=1}^m \|\mathbf{c}_i^0 - \mathbf{c}^*\|_2^2 - L_d \sum_{t=1}^T \sum_{i=1}^m \|\bar{\mathbf{d}}^t - \mathbf{d}_j^t\| - \frac{\eta_2}{2} m T L_d^2
\end{aligned}$$

Notice f is convex for $\mathbf{v}$, we have $f(\mathbf{v}_i^t, \mathbf{c}^*) \geq f(\mathbf{v}^*, \mathbf{c}^*)$, and therefore

$$\begin{aligned}
&\sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{v}_i^t, \bar{\mathbf{d}}^t) - f(\mathbf{v}^*, \mathbf{c}^*)] \\
&\geq -\frac{1}{2\eta_2} \sum_{i=1}^m \|\mathbf{c}_i^0 - \mathbf{c}^*\|_2^2 - L_d \sum_{t=1}^T \sum_{i=1}^m \|\bar{\mathbf{d}}^t - \mathbf{d}_j^t\| - \frac{\eta_2}{2} m T L_d^2.
\end{aligned}$$

This concludes the proof. $\square$

4.5.5 Proof for Eq (4.15)

In the previous research, [Ying et al.](#) have proved online AUC maximization is equivalent to Eq (4.15) if ℓ_2 loss is adopted. For the completeness of this part, we shall provide the proof below.

The general AUC maximization can be written as

$$\arg \min_{\hat{\mathbf{w}} \in \mathcal{W}} \mathbb{E} [(1 - \hat{\mathbf{w}}^\top (\mathbf{x} - \mathbf{x}')^2 \mid y = 1, y' = -1] .$$

Rewrite above formula as

$$\begin{aligned} & \mathbb{E} [(1 - \hat{\mathbf{w}}^\top (\mathbf{x} - \mathbf{x}')^2 \mid y = 1, y' = -1] \\ &= 1 + \mathbb{E} [(\hat{\mathbf{w}}^\top \mathbf{x})^2 \mid y = 1] + \mathbb{E} [(\hat{\mathbf{w}}^\top \mathbf{x}')^2 \mid y' = -1] \\ & \quad - 2\mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x} \mid y = 1] + 2\mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x}' \mid y' = -1] \\ & \quad - 2(\mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x} \mid y = 1])(\mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x}' \mid y' = -1]) \\ &= 1 + \underbrace{\mathbb{E} [(\hat{\mathbf{w}}^\top \mathbf{x})^2 \mid y = 1] - (\mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x} \mid y = 1])^2}_{\textcircled{1}} \\ & \quad + \underbrace{\mathbb{E} [(\hat{\mathbf{w}}^\top \mathbf{x}')^2 \mid y' = -1] - (\mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x}' \mid y' = -1])^2}_{\textcircled{2}} \\ & \quad - 2\mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x} \mid y = 1] + 2\mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x}' \mid y' = -1] \\ & \quad + \underbrace{(\mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x} \mid y = 1] - \mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x}' \mid y' = -1])^2}_{\textcircled{3}} \end{aligned}$$

Denote $\mathbf{z} = (\mathbf{x}, y)$ as a random sample draw from a distribution $\mathcal{Z}$, we have

$$\begin{aligned} \textcircled{1} &= \frac{1}{p} \int_{\mathcal{Z}} (\hat{\mathbf{w}}^\top \mathbf{x})^2 \mathbb{I}_{[y=1]} d\rho(z) - \left(\frac{1}{p} \int_{\mathcal{Z}} \hat{\mathbf{w}}^\top \mathbf{x} \mathbb{I}_{[y=1]} d\rho(z) \right)^2 \\ &= \min_{\beta \in \mathbb{R}} \frac{1}{p} \int_{\mathcal{Z}} (\hat{\mathbf{w}}^\top \mathbf{x} - \beta)^2 \mathbb{I}_{[y=1]} d\rho(z) \\ &= \min_{\beta \in \mathbb{R}} \mathbb{E} [(\hat{\mathbf{w}}^\top \mathbf{x} - \beta)^2 \mid y = 1] \end{aligned}$$

where the minimization is achieved by $\beta = \mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x} \mid y = 1]$.

Similarly,

$$\textcircled{2} = \min_{\gamma \in \mathbb{R}} \mathbb{E} [(\hat{\mathbf{w}}^\top \mathbf{x}' - \mathbf{g})^2 \mid y' = -1],$$

where the minimization is achieved by $\gamma = \mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x}' \mid y' = -1]$.

$$\textcircled{3} = \max_{\alpha \in \mathbb{R}} \{2\alpha (\mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x}' \mid y' = -1] - \mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x} \mid y = 1]) - \alpha^2\}$$

where the maximization is achieved by $\alpha = \mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x}' \mid y' = -1] - \mathbb{E} [\hat{\mathbf{w}}^\top \mathbf{x} \mid y = 1]$.

Putting all these equations back, we have

$$\begin{aligned} & \mathbb{E} [(1 - \hat{\mathbf{w}}^\top (\mathbf{x} - \mathbf{x}'))^2 \mid y = 1, y' = -1] \\ &= 1 + \mathbb{E} [(\hat{\mathbf{w}}^\top \mathbf{x})^2 \mid y = 1] + \mathbb{E} [(\hat{\mathbf{w}}^\top \mathbf{x}')^2 \mid y' = -1] \\ &= 1 + \frac{1}{p(1-p)} \min_{\substack{\hat{\mathbf{w}} \in \mathcal{W} \\ (\beta, \gamma) \in \mathbb{R}^2}} \max_{\alpha \in \mathbb{R}} \int_{\mathcal{Z}} F(\hat{\mathbf{w}}, \beta, \mathbf{g}, \alpha; z) d\rho(z), \end{aligned}$$

with

$$\begin{aligned} F(\hat{\mathbf{w}}, \beta, \mathbf{g}, \alpha; z) &= (1-p)(\hat{\mathbf{w}}^\top \mathbf{x} - \beta)^2 \mathbb{I}_{[y=1]} \\ &+ p(\hat{\mathbf{w}}^\top \mathbf{x} - \mathbf{g})^2 \mathbb{I}_{[y=-1]} - p(1-p)\alpha^2 \\ &+ 2(1+\alpha) (p\hat{\mathbf{w}}^\top \mathbf{x} \mathbb{I}_{[y=-1]} - (1-p)\hat{\mathbf{w}}^\top \mathbf{x} \mathbb{I}_{[y=1]}). \end{aligned}$$

This concludes the proof. □

Chapter 5

Online Federated Learning for Decentralized Multi-task Optimization

In the previous two chapters, we focus on reaching a common minimizer by sequentially using local adaptation and network diffusion. An underlying mechanism for this kind of online federated learning is that a single task is deployed on the graphs and there exists the same optimal value for all nodes. In this chapter, we will look into the scenarios and seek solutions for decentralized multi-task learning.

Although dispersing one single task (STL) to distributed learning nodes has been intensively studied by the previous research, multi-task learning (MTL) on distributed networks is still an area that has not been fully exploited, especially under decentralized settings. The challenge lies in the fact that different tasks may have different optimal learning weights while communication through the distributed network forces all tasks to converge to a unique classifier.

In this chapter, we present a novel algorithm to overcome this challenge and enable learning multiple tasks simultaneously on a decentralized distributed network. Specifically, the learning framework can be separated into two phases: (i) multi-task information is shared within each node on the first phase; (ii) communication between nodes then leads the whole network to converge to a common minimizer. Theoretical analysis indicates that our algorithm achieves an $\mathcal{O}(\sqrt{T})$ regret bound when compared with the

best classifier in hindsight, which is further validated by experiments on both synthetic and real-world datasets.

5.1 Motivations and Existing Solutions

Multi-task learning (MTL) aims at improving the generalization performance by learning multiple tasks in parallel and exploiting their intrinsic relationship. While single-task learning (STL) provides a simple solution to learn each task independently, in many cases, data from multiple sources may share similar patterns and can jointly contribute to improving each other's learning performance. A typical example can be found in the market where feedback from different users can be closely related and shares similar traits, and an efficient learning algorithm should be able to depict the relatedness of collected data. Representative MTL methods to boost the generalization performance on batch data include: common parameter sharing [70], low-rank subspace projection [41, 71] and trace norm regularization [72]. On the other side, recent years also witness extensive studies [45, 46, 73–75] on streaming data, also known as online multi-task learning (OMTL) algorithms, for the merits of low computation and memory requirement and strong adaptation to the changing environment.

The big-data era has brought new challenges for MTL since it naturally bears a mission of computing multiple tasks simultaneously, and it's more desirable to distribute computation that is locally light. In many scenarios, data may be collected from different places and sending all data back to one learning center obviously leads to extra transportation cost and is also limited to geometric distance. It's apparently more favorable to distribute smaller subsets of data to different learning units and then collaborate with direct neighbors to find the optimal solution [12]. Along with these advantages of distributed learning algorithms also comes the merits as link or node failure insensitivity and user's privacy protection.

Despite these aforementioned benefits and its strong demand, distributed MTL is still an area that has not been fully exploited. Dinuzzo [76] first explored distributed MTL by performing information fusion with a server-client strategy, while later research

Figure 5.1: Two tasks with different optimal value ω_1^* and ω_2^* . The network will finally converge to some point ω between ω_1^* and ω_2^* . Obviously, any attempt to decrease one task's loss lies at the cost of increasing the other's (e.g., moving ω down along the red line increases $J_2(\omega)$).

[77, 78] proposed distributed MTL algorithms by further restricting task relatedness through shared sparsity. They are all under centralized setting and use the common ‘master and worker’ framework, where the ‘master’ collects real-time information from all tasks to construct a global model and then feeds back to each ‘worker’ to further exploit the local content.

MTL under decentralized settings is more challenging. First of all, each node only communicates with its direct neighbors and there is no global model (‘master’) on decentralized distributed networks. Direct information flow is strictly restricted within a limited range, and any attempt to construct global sparsity is impractical. On the other hand, previous research [12] has proved that all nodes will finally converge to the common optimal value after sufficient training, but this property brings a more challenging fact that the optimal solutions for different tasks may just be similar rather than identical. This discrepancy is known as **‘Pareto optimality’** between each task, which is actually the compromise of reaching a global optimality at the cost of sacrificing each task's own loss. Figure 5.1 gives a simple illustration for ‘Pareto optimality’ of two tasks, where the final ω terminates at some point between ω_1^* and ω_2^* . Apparently, there is no guarantee such a compromise will be better than learning each task individually without communication.

In order to overcome these challenges, Chen [79] presented a possible solution by

clustering similar tasks into groups and only required tasks within each group to share the same minimizer, while another possible solution [80, 81] is assuming that the node-specific parameters lie in a common subspace. However, these methods all assume that we have some prior knowledge beforehand, such as how the parameter space is divided or which cluster each task belongs to, and may not hold in real-world scenarios. Besides, these methods strictly rely on the similarity of tasks and are very sensitive to outlier tasks.

Motivated by the merits and challenges of distributed MTL, we present a new learning framework and jump out of the stereotype of limiting each learning node to a ‘worker’ for a specific task. A more general setting is therefore proposed where each learning unit is able to learn all tasks, and knowledge is transferred in two phases: task information is shared within each node in the first phase, and then all nodes collaborate together to achieve a common minimizer. The proposed algorithms successfully avoid suffering the challenge of ‘Pareto optimality’ since the potential optimum for all nodes is identical rather than similar.

Our key contribution lies in two aspects. The first is to propose a simple but novel framework for decentralized distributed OMTL problem, which is not only applicable to our own algorithms but also able to transform many existing distributed STL methods (e.g. distributed dual averaging algorithm [37]) into decentralized distributed MTL algorithms. To the best of our knowledge, this is the first model to enable OMTL on decentralized distributed networks without any prior knowledge. The second contribution is that our algorithm enjoys a lower $\mathcal{O}(\sqrt{T})$ theoretical regret if given a fixed relationship matrix, while previous studies either only provides empirical results [46] or be upper bounded with a larger regret under distributed setting [See $R_T = \mathcal{O}(\sqrt{T} \log T)$ of theorem 2 in [37]].

5.2 Online Federated Learning for Decentralized Multi-task Optimization (DOM)

We will present our new learning framework in the section, together with two corresponding algorithms. The novel part is that we separate information flow into two phases: transferring multi-task information within each node first and then collaborate with direct neighbors to achieve a common minimizer on the network.

5.2.1 Problem Formulation

Considering a set of m tasks for classification in parallel and each task receives one instance $\mathbf{x}_t^i \in \mathbb{R}^d$ on each round, OMTL algorithm is required to make predictions for each task based on its stored learning weight W_t : $\hat{y}_t^i = \text{sign}(\hat{q}_t^i) = \text{sign}((\mathbf{w}_t^i)^\top \mathbf{x}_t^i)$ for $i \in [1, \dots, m]$, where $\mathbf{w}_t^i$ refers to the i -th column of W_t . Its predictions are evaluated by a canonical loss function f , and in this chapter we adopt the hinge loss $f = [1 - y_t^i(\mathbf{w}_t^i)^\top \mathbf{x}_t^i]_+$ for simplicity. Upon receiving the penalty, the algorithm updates $\mathbf{w}_{t+1}$ to minimize its current hinge loss and prepares for predictions on the next round. Our ultimate goal is minimizing the overall learning loss compared with the best $\mathbf{w}_{i,*}$ for task i in hindsight [13], also known as *regret*,

$$R_T = \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{w}_{i,t}) - f(\mathbf{w}_{i,*})].$$

Instead of assigning each node to tackle one specific task [79–81] and encountering the compromise of reaching ‘Pareto optimality’, we propose a more general setting where each node is able to deal with all tasks and instances from different tasks can be randomly allocated to any learning node.

Specifically, each node concatenates instances from all m tasks as a compound instance vector

$$\phi_{i,t}^\top = [(\mathbf{x}_{i,t}^1)^\top, (\mathbf{x}_{i,t}^2)^\top, \dots, (\mathbf{x}_{i,t}^m)^\top], \forall i \in [1, \dots, m]$$

and the corresponding weight vector for the i -th node is compounded as

$$\mathbf{v}_{i,t}^\top = [(\mathbf{w}_{i,t}^1)^\top, (\mathbf{w}_{i,t}^2)^\top, \dots, (\mathbf{w}_{i,t}^m)^\top], \forall i \in [1, \dots, m]. \quad (5.1)$$

For the convenience of analysis, we denote $\mathbf{w}_*^i$ as the optimal learning weight for the i -th task and $\mathbf{v}_{i,*}$ as the optimal learning weight for the i -th node. If following the stereotype of assigning each node to be a ‘worker’ for a specific task, the optimal weight of any learning node will be equal to that of its assigned task, namely $\mathbf{v}_{i,*} = \mathbf{w}_*^i$. Since $\mathbf{w}_*^i$ may differ from each other, the optimal weights on different nodes will also be disparate and can’t be efficiently learned from the networks.

On the other hand, the optimal value $\mathbf{v}_{i,*}^\top = [(\mathbf{w}_*^1)^\top, (\mathbf{w}_*^2)^\top, \dots, (\mathbf{w}_*^m)^\top]$ is identical for all nodes under our proposed learning framework, and it also endows each node with the ability to learn all tasks other than being limited to a certain task. These are the insights into our methodology of concatenating learning weights as a compound vector in this part.

5.2.2 Node Update

Our assumption is that the algorithm receives m instances and randomly allocates them to m learning nodes¹ on each round. Suppose the i -th node receives an instance $\mathbf{x}_t^{i_t}$ from the i_t -th task, and the compound instance vector is

$$\phi_{i,t}^\top = [0, \dots, (\mathbf{x}_t^{i_t})^\top, \dots, 0]. \quad (5.2)$$

The prediction becomes

$$\hat{y}_{i,t} = \text{sign}(\hat{q}_{i,t}) = \text{sign}(\mathbf{v}_{i,t}^\top \phi_{i,t}) = \text{sign}((\mathbf{w}_{i,t}^{i_t})^\top \mathbf{x}_t^{i_t}). \quad (5.3)$$

¹In the remaining part, we simply assume there are m nodes on the network, but it’s applicable to the case where there are more nodes than instances by updating $\nabla f = 0$ on nodes that do not receive instances.

Evolution on the network is apparently inefficient if we simply update i_t -th task's learning weight as

$$\mathbf{w}_{i,t+1}^{i_t} = \mathbf{w}_{i,t}^{i_t} - \eta \nabla f(y_{i,t}, \hat{q}_{i,t}). \quad (5.4)$$

We need to design a proper scheme to share the instance information for all tasks within each node.

5.2.2.1 Node updating rule

Following the research in [82], we define the multi-task kernel feature mapping Ψ so that $x_t^{i_t} \in \mathbb{R}^d$ is projected into md -dimension Reproducing Kernel Hilbert Space (RKHS) $\mathcal{H}$ such that $\Psi(x_t^{i_t}) = B_{\otimes}^{-1} \phi_t$, and then the corresponding inner product is defined as

$$\langle \Psi(x_s^{i_s}), \Psi(x_t^{i_t}) \rangle_{B_{\otimes}} = \left(B_{\otimes}^{-1} \phi_s \right)^{\top} B_{\otimes} \left(B_{\otimes}^{-1} \phi_t \right) = \phi_s^{\top} B_{\otimes}^{-1} \phi_t,$$

where B refers to a $m \times m$ symmetric matrix which captures the *relationship* among tasks, and $B_{\otimes}$ refers to the Kronecker product $B_{\otimes} = B \otimes I_d$.

The hinge loss function in the multi-task kernel space is then defined as

$$f(\mathbf{v}) = \max \left\{ 0, 1 - y_t^{i_t} \langle \mathbf{v}, \Psi(x_t^{i_t}) \rangle_{B_{\otimes}} \right\},$$

and if $f(\mathbf{v})$ is not zero, the derivative will be

$$\nabla f(\mathbf{v}) = -y_t^{i_t} \Psi(x_t^{i_t}) = -y_t^{i_t} B_{\otimes}^{-1} \phi_t. \quad (5.5)$$

For node i , our goal is selecting the optimal $\mathbf{v}_i$ to minimize its current loss, or

$$\min_{\mathbf{v}_i} f(\mathbf{v}_i)$$

With Taylor's expansion in the kernel space, we replace such a minimization problem with

$$\min_{\mathbf{v}_i} \langle \nabla f, \mathbf{v}_i \rangle_{B_\otimes} + \frac{1}{2\eta} \|\mathbf{v}_i - \mathbf{v}_{i,t}\|_{B_\otimes}^2.$$

Setting the derivative to zero, we have the updating rule as

$$\mathbf{v}_i = \mathbf{v}_{i,t} - \eta \nabla f(\mathbf{v}_{i,t}). \quad (5.6)$$

Substituting $\nabla f(\mathbf{v}_{i,t})$ with Eq (5.5), the updating rule for each node as

$$\mathbf{v}_i = \mathbf{v}_{i,t} + \eta y_t^{i_t} B_\otimes^{-1} \phi_{i,t}. \quad (5.7)$$

Recalling the equation $B_\otimes^{-1} = B^{-1} \otimes I_d$ and the expressions for $\mathbf{v}_i$ and $\phi_{i,t}$ in Eq (5.1) (5.2), we have the closed-form updating rule for j -th task as

$$\mathbf{w}_{i,t+1}^j = \mathbf{w}_{i,t}^j + \eta B_{j,i_t}^{-1} \mathbf{x}_t^{i_t} y_t^{i_t} \quad \text{for } \forall j \in [1, \dots, m], \quad (5.8)$$

where B_{j,i_t}^{-1} denotes the (j, i_t) -th component of matrix B^{-1} . Compared with Eq (5.4) where only the i_t -th task is updated, Eq (5.8) updates learning weights for all tasks by transferring information in the multi-task kernel space.

5.2.2.2 Selection of relationship matrix B

For the selection of matrix B, we first present an algorithm based on a fixed relationship matrix [45],

$$B_{j,i_t}^{-1} = \frac{b + m\delta_{j,i_t}}{(b+1)m}, \quad (5.9)$$

where $\delta_{j,i_t} = 1$ if $j = i_t$ and $\delta_{j,i_t} = 0$ otherwise. The corresponding updating rule for each task becomes

$$\mathbf{w}_{i,t+1}^j = \mathbf{w}_{i,t}^j + \eta \frac{b + m\delta_{j,i_t}}{(b+1)m} y_t^{i_t} \mathbf{x}_t^{i_t} \quad \text{for } \forall j \in [1, \dots, m]. \quad (5.10)$$

Algorithm 5 Decentralized Distributed Online Multi-task Classification with Fixed B (DOM-I)

```

1: Initialize:  $a_{i,j} = 1/N$  in matrix  $A$ , and  $B^{-1}$  as Eq (5.9).
2: Initialize:  $\mathbf{u}_{i,0} = \mathbf{0}$  for  $\forall i \in [1, \dots, m]$ ;
3: for  $t = 0, \dots, T - 1$  do
4:   for  $i = 1, \dots, m$  do
5:     Combine:  $\mathbf{v}_{i,t} = \sum_{j=1}^m a_{i,j} \mathbf{u}_{j,t}$ 
6:   end for
7:   for  $i = 1, \dots, m$  do
8:     Receive instance pair  $(\mathbf{x}_t^{i_t}, y_t^{i_t})$ 
9:     Calculate  $\hat{q}_{i,t} = \mathbf{v}_{i,t} \cdot \mathbf{x}_t^{i_t}$ 
10:    Predict  $\hat{y}_{i,t} = \text{sign}(\hat{q}_{i,t})$ 
11:    Compute the loss function  $f(\mathbf{v}_{i,t}) = [1 - y_{i,t} \hat{q}_{i,t}]_+$ 
12:    if  $f(\mathbf{v}_{i,t}) > 0$  then
13:      Update  $\mathbf{u}_{i,t+1} = \mathbf{v}_{i,t} + \eta y_t^{i_t} B_{\otimes}^{-1} \phi_{i,t}$ 
14:    else
15:       $\mathbf{u}_{i,t+1} = \mathbf{v}_{i,t}$ 
16:    end if
17:  end for
18: end for
19: Output:  $\mathbf{u}_i$  for  $i = 1, \dots, m$ 

```

The intuition behind Eq (5.10) is that the tasks' relationship is reflected by parameter b , and the optimal b_* will be given in section III or simply be tuned on real-world datasets. Specifically, if $b = 0$, there is no connection behind tasks and only i_t -th task will be updated, while if $b = \infty$, all tasks share the same information of instance $(x_t^{i_t}, y_t^{i_t})$ with the weight $\frac{1}{m}$.

Since the relationship between tasks may not be static in the long run, we also propose a dynamic relationship matrix B with a smooth updating rule [83]¹,

$$B_{i,t}^{-1} = \alpha B_0^{-1} + (1 - \alpha) P_{i,t}, \quad (5.11)$$

where B_0^{-1} refers to an initial matrix that can be set to be I or any fixed row stochastic matrix and P_t is the relationship matrix learned from data on each round. Specially, the

¹Although dynamic relationship seems more meaningful in many scenarios, we can't confirm B is p.s.d and hence this methods leads to the challenge of theoretical analysis [46].

Algorithm 6 Decentralized Distributed Online Multi-task Classification with Dynamic B (DOM-II)

```

1: Initialize:  $a_{i,j} = 1/N$  in matrix  $A$ .
2: Initialize:  $\mathbf{u}_{i,0} = \mathbf{0}$ ,  $P_{i,0} = 0$  for  $\forall i \in [1, \dots, m]$ 
3: Initialize:  $\hat{B}_{i,0} = \alpha B_0^{-1}$  for  $\forall i \in [1, \dots, m]$ 
4: for  $t = 0, \dots, T - 1$  do
5:   for  $i = 1, \dots, m$  do
6:     Combine:  $\mathbf{v}_{i,t} = \sum_{j=1}^m a_{i,j} \mathbf{u}_{j,t}$ 
7:     Combine:  $B_{i,t+1} = \sum_{j=1}^m a_{i,j} \hat{B}_{j,t}$ 
8:   end for
9:   for  $i = 1, \dots, m$  do
10:    Receive instance pair  $(\mathbf{x}_t^{i_t}, y_t^{i_t})$ 
11:    Calculate  $\hat{q}_{i,t} = \mathbf{v}_{i,t} \cdot \mathbf{x}_t^{i_t}$ 
12:    Predict  $\hat{y}_{i,t} = \text{sign}(\hat{q}_{i,t})$ 
13:    Compute the loss function  $f(\mathbf{v}_{i,t}) = [1 - y_{i,t} \hat{q}_{i,t}]_+$ 
14:    if  $f(\mathbf{v}_{i,t}) > 0$  then
15:      Update  $\mathbf{u}_{i,t+1} = \mathbf{v}_{i,t} + \eta y_t^{i_t} [B_{i,t+1}^{-1}] \otimes \phi_{i,t}$ 
16:    else
17:       $\mathbf{u}_{i,t+1} = \mathbf{v}_{i,t}$ 
18:    end if
19:    Update  $P_{i,t+1}$  as Eq (5.12)
20:    Update  $\hat{B}_{i,t+1} = \alpha B_0^{-1} + (1 - \alpha) P_{i,t+1}$ 
21:  end for
22: end for
23: Output:  $\mathbf{u}_i$  for  $i = 1, \dots, m$ 

```

(i_t, j) -th component of $P_{i,t}$ will be updated as

$$p_{i_t,j} = \frac{\exp(-f(\mathbf{w}_{j,t}^{i_t}, \mathbf{x}_t^{i_t}, y_t^{i_t}))}{\sum_{j'=1}^m \exp(-f(\mathbf{w}_{j',t}^{i_t}, \mathbf{x}_t^{i_t}, y_t^{i_t}))}. \quad (5.12)$$

The intuition behind Eq (5.12) is that if the $\mathbf{w}_{j,t}^{i_t}$ also works well for $(\mathbf{x}_t^{i_t}, y_t^{i_t})$, it will be assigned with a large transferring weight coefficient from j to i_t , and the normalization ensures $P_{i,t}$ is still a row stochastic matrix after updating.

5.2.3 Nodes Communication

After designing multi-task information sharing scheme within each node, we now turn to the investigation of communication among the nodes. We assume that the network

topology is *static* and information among nodes are exchanged through a *fixed* combination matrix A . Since the previous research [84] shows that diffusion learning outperforms incremental learning and consensus learning, we adopt diffusion method with the strategy ‘Combine and Then Adapt’ (CTA) in this thesis.

Denote weight vector for the i -th node at the end of round t as $\mathbf{u}_{i,t}$, then our updating rule is:

$$\text{Step I : } \mathbf{v}_{i,t} = \sum_{j=1}^m a_{i,j} \mathbf{u}_{j,t}, \quad (5.13)$$

$$\text{Step II : } \mathbf{u}_{i,t+1} = \mathbf{v}_{i,t} - \eta \nabla f(\mathbf{v}_{i,t}), \quad (5.14)$$

where $a_{i,j}$ denotes the (i, j) -th component of combination matrix A . Node i first combines learning weights from its direct neighbors and then repeats Eq (5.6) to perform an update by learning from the newly-come instance in the multi-task kernel space.

The combination weight matrix A , together with its components $a_{i,j}$, determine how information flows within the network. In order to guarantee efficient updating, we make a few assumptions below.

Assumption 5.1. Denote i -th node’s direct neighbors as $N(i)$ and the number of $N(i)$ as N . If $j \in N(i)$, there exists a scalar ϵ such that $a_{i,j} \geq \epsilon$ holds. If $j \notin N(i)$, $a_{i,j} = 0$ holds.

Assumption 5.2. $\sum_{j=1}^m a_{i,j} = 1$ and $\sum_{i=1}^m a_{i,j} = 1$ holds for any i and j .

Assumption 5.3. There exists at least one node such that $i \in N(i)$, and there does not exist any j such that $N(j) = \{j\}$.

Assumption 5.1 ensures each node gives sufficient learning weight to any of its direct neighbors and does not combine learning weights from indirect neighbors. Assumption 5.2 guarantees the network is doubly stochastic, and Assumption 5.3 confirms that there is no separated node.

Finally, algorithm 5 and 6 summarize our decentralized distributed online multi-task classification algorithms with a fixed B and a dynamic B separately.

5.3 Theoretical Analysis

We analyze the regret bound for DOM with a fixed relationship matrix in this part.

5.3.1 Proof Setup

We start by defining several auxiliary notations to help us track the information flow within the network.

There is no ‘external’ information flowing into the network in step I since all nodes simply combine their neighbors’ weight. Step II introduces a new instance for each node, together with its gradient information. In order to capture the information, we define

$$\mathbf{p}_{i,t+1} = \mathbf{u}_{i,t+1} - \mathbf{v}_{i,t} \stackrel{(5.14)}{=} -\eta \nabla f(\mathbf{v}_{i,t}).$$

Assume that all the instances are within the space $\Psi : \{\mathbf{x} \mid \|\mathbf{x}\| \leq D\}$, we have the upper bound of $\mathbf{p}$ as

$$\|\mathbf{p}_{i,t+1}\|_{B_{\otimes}} \stackrel{(5.5)}{=} \|\eta y_t^{i_t} B_{\otimes}^{-1} \phi_{i,t}\|_{B_{\otimes}} \stackrel{(5.2)(5.9)}{=} \eta \|\mathbf{x}_t^{i_t}\| \sqrt{(B^{-1})_{i_t, i_t}}.$$

Denoting $L = \|\mathbf{x}_t^{i_t}\| \sqrt{(B^{-1})_{i_t, i_t}}$, information flow is therefore bounded as

$$\|\mathbf{p}_{i,t+1}\|_{B_{\otimes}} \leq \eta L \tag{5.15}$$

Since any two norms on finite-dimensional vector space are equivalent [85] and assuming B is p.s.d, we have

$$c \|\nabla f(\mathbf{v}_{i,t})\|_{B_{\otimes}} \leq \|\nabla f(\mathbf{v}_{i,t})\|_{B_{\otimes}^{-1}} \leq C \|\nabla f(\mathbf{v}_{i,t})\|_{B_{\otimes}}, \tag{5.16}$$

for some constant c, C .

On the other hand, a matrix chain is defined as $\Phi(k, s) = A^{k-s}$ ($k \geq s \geq 0$) so the evolution of $\mathbf{u}_{j,t+1}$ is derived as

$$\mathbf{u}_{j,t+1} = \mathbf{p}_{j,t+1} + \sum_{l=1}^t \sum_{i=1}^m [\Phi(t+1, l)]_{j,i} \mathbf{p}_{i,l}. \quad (5.17)$$

We define an auxiliary ‘central’ weight as $\mathbf{z}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{u}_{i,t}$, and its evolution can be derived as

$$\mathbf{z}_t = \frac{1}{m} \sum_{i=1}^m \mathbf{u}_{i,0} + \frac{1}{m} \sum_{l=1}^t \sum_{i=1}^m \mathbf{p}_{i,l}$$

Without loss of generality, we assume all the learning weights start from $\mathbf{0}$, or $\mathbf{u}_{i,0} = \mathbf{0}$. Then the above equation can be written as

$$\mathbf{z}_t = \frac{1}{m} \sum_{l=1}^t \sum_{i=1}^m \mathbf{p}_{i,l}. \quad (5.18)$$

5.3.2 Weight Difference and Regret Bound

Our first goal is deriving the difference between these two learning weights, or $\|\mathbf{z}_t - \mathbf{u}_{j,t}\|_{B_\otimes}$.

Lemma 5.4. *Suppose the network is static and strongly connected, and assumptions 1,2,3 hold. We have the following inequality:*

$$|[\Phi(k, s)_{i,j} - \frac{1}{m}]| \leq \theta \beta^{k-s}$$

for all $k \geq s \geq 0$, where $\theta = (1 - \frac{\epsilon}{4m^2})^{-2}$ and $\beta = (1 - \frac{\epsilon}{4m^2})^{\frac{1}{Q}}$ only depend on the size and the topology of the network.

The above lemma numerically calculates how fast the information flows into separate nodes before the uniform distribution, and enables our further analysis on bounding the weight difference in the following lemma.

Lemma 5.5. *Consider all the instances are within the range Ψ and Lemma 5.4 holds, then we have*

$$\|\mathbf{z}_{t+1} - \mathbf{u}_{j,t+1}\|_{B_{\otimes}} \leq \eta L \Delta$$

where $\Delta = m\theta \frac{\beta}{1-\beta} + 2$.

Equipped with these two lemmas, we can march towards the second goal of minimizing the regret between sequence $\{\mathbf{u}_{i,t}\}$ and the potential best classifier $\mathbf{u}_*$, or

$$\min \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{u}_{i,t}) - f(\mathbf{u}_*)]. \quad (5.19)$$

Given an arbitrary $\mathbf{u}$, we have

$$\begin{aligned} & \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{u}_{i,t}) - f(\mathbf{u})] \\ &= \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] + \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{u}_{i,t}) - f(\mathbf{z}_t)] \\ &\leq \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] + \sum_{t=1}^T \sum_{i=1}^m \|\nabla f(\mathbf{v}_{i,t})\|_{B_{\otimes}^{-1}} \|\mathbf{u}_{i,t} - \mathbf{z}_t\|_{B_{\otimes}} \\ &\stackrel{(5.16)}{\leq} \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] + \sum_{t=1}^T \sum_{i=1}^m CL \|\mathbf{u}_{i,t} - \mathbf{z}_t\|_{B_{\otimes}} \end{aligned} \quad (5.20)$$

Eq (5.20) includes two terms, and we derive the upper bound for the first term in the following lemma.

Lemma 5.6. *The difference between $\mathbf{z}_t$ and any fixed $\mathbf{u}$ is bounded as*

$$\sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] \leq \frac{m\|\mathbf{u}\|_{B_{\otimes}}^2}{2\eta} + CL \sum_{t=1}^T \sum_{i=1}^m \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_{\otimes}} + \eta m T L^2 / 2.$$

Combining Lemma 5.5 and Lemma 5.6 into Eq (5.20), we come to the following theorem.

Theorem 5.7. Let $\mathbf{u}_*$ denotes the optimal classifier in hindsight, then the regret of the proposed DOM algorithm is upper bounded as

$$\sum_{t=1}^T \sum_{i=1}^m f(\mathbf{u}_{i,t}) - f(\mathbf{u}_*) \leq \frac{m \|\mathbf{u}_*\|_{B_\otimes}^2}{2\eta} + \eta m T L^2 \left(\frac{1}{2} + 2C\Delta \right).$$

Remark: (1) By selecting $\eta = \frac{\|\mathbf{u}_*\|_{B_\otimes}}{L\sqrt{T(1+4C\Delta)}}$, the optimum regret bound can be written as

$$mL\sqrt{(T+4C\Delta)\|\mathbf{u}_*\|_{B_\otimes}^2},$$

which is actually a $\mathcal{O}(\sqrt{T})$ regret.

(2) The regret bound depends on the relationship between the optimum weight $\mathbf{u}_*^i$ of each task. If B is static, then

$$\|\mathbf{u}_*\|_{B_\otimes}^2 = \sum_{i=1}^m \|\mathbf{u}_*^i\|^2 + b \sum_{i=1}^m \|\mathbf{u}_*^i - \bar{\mathbf{u}}\|^2,$$

where $\bar{\mathbf{u}} = \frac{1}{m} \sum_{i=1}^m \mathbf{u}_*^i$ is the average of all potential optimal weight vectors. By the definition of L and $\|\mathbf{u}_*\|_{B_\otimes}^2$ and if $\|\mathbf{u}_*^i\|^2 < \frac{m}{m-1} \|\mathbf{u}_*^i - \bar{\mathbf{u}}\|^2$, the optimal b_* can be derived as

$$b_* = \sqrt{(m-1) \frac{\sum_{i=1}^m \|\mathbf{u}_*^i\|^2 - \sum_{i=1}^m \|\mathbf{u}_*^i - \bar{\mathbf{u}}\|^2}{\sum_{i=1}^m \|\mathbf{u}_*^i - \bar{\mathbf{u}}\|^2}},$$

or $b = 0$ otherwise.

(3) Decentralized network with good connection property leads to a small Δ (since θ and β are relatively small), and therefore its learning regret is also lower compared with poorly connected network.

5.4 Experiment

In this section, we compare our algorithms with several competitors on both synthetic and real-world datasets to explore the influence of multi-task learning and distributed learning. We start by introducing our experiment data and comparison algorithms, followed by discussions on the results.

5.4.1 Benchmarks

Our work highlights the challenges of distributing OMTL to multiple nodes, and here we classify all competing algorithms into two groups:

1. Centralized Learning:

- **PM**: Original Perceptron algorithm [14] by adding a learning step to adjust its adaptation to data.
- **PA**: Online Passive-Aggressive algorithm [16].
- **COML**: Collaborative Online Multi-task Learning [74] represent centralized OMTL method.

2. [topsep=0.2ex,itemsep=0.2ex,partopsep=0.2ex,parsep=0.2ex]

NS: Previous research [79] of assigning each node to tackle one specific task, also known as **Node Specific** methods.

- **DST**: **D**istributed **S**ingle-**T**ask method disperses each task to all learning nodes, and there is no information sharing between tasks.
- **DA**: **D**ual **A**veraging algorithm for distributed optimization [37] is originally designed for single-task learning, but it can also be extended to OMTL setting under our presented framework.
- **DOM-I**: The proposed algorithm with a fixed relationship matrix. Each node can tackle all tasks, and the information is shared for all tasks and nodes.

- **DOM-II**: Learning with a smooth dynamic relationship matrix.

Three different types of networks, namely full-connected, grid-connected and ring-connected network, are used to examine the effect of network topology. Each learning node equally assigns $1/N$ weights to its direct neighbors, where N represents the number of its direct neighbors. The performance of these algorithms is evaluated by their error prediction rates, and each experiment consists of 50 runs, where the instances are shuffled beforehand and randomly assigned to the learning nodes. The learning rate η is tuned within a grid search $\{10^{-5}, \dots, 10^{+2}\}$.

We first construct a **synthetic** dataset via random walks in parameter space with Gaussian increments. The initial value is $\mathbf{w}^0 \in \mathbb{R}^{100}$, and its first 60 components are set to be 1 while the rest are set to be -1.5 . We then construct the potential optimal weight vector for 9 similar tasks as $\mathbf{w}^{i+1} = \mathbf{w}^i + \varepsilon$, where $\varepsilon \sim N(0, 0.1I)$ and $i = 0, \dots, 8$. A training sample $\mathbf{x} = [x_1, \dots, x_{100}]$ is generated by randomly selecting $x_i \in [-3, 3]$, and for each task we obtain 2000 samples.

Algorithm efficiency is tested also on two commonly used real-world datasets: **MHC-I**¹ [86] contains a subset of human MHC-I alleles (A0201, A0202, A0203, A0206, A0301, A3101, A3301, A6801, A6802) and features are extracted with bigram amino acid encoding [74] to project each protein sequence into 400-dimension feature space. **Sentiment**² [87] contains user reviews of 4 types of products (books, DVD, electronics and kitchen) and each review lies in the 230610-dimension feature space based on the corresponding word sequence.

5.4.2 Performance Analysis

The synthetic dataset consists of 9 similar tasks, and from Table 5.1 we can observe that OMTL strategies significantly improve the overall performance, reducing the error rate from 12% to 6%. Previous studies have validated that OMTL algorithms boost the

¹<http://tools.iedb.org/main/datasets/>

²Sentiment dataset contains 4 tasks and ring-connected network is the same as grid-connected network.

Method	Synthetic	MHC-I	Sentiment
Centralized			
PM	12.95 ± 0.19	28.22 ± 0.21	22.06 ± 0.44
PA	12.66 ± 0.19	28.09 ± 0.15	21.85 ± 0.37
COML	6.00 ± 0.14	27.58 ± 0.16	21.13 ± 0.32
Decentralized			
NS (Full)	6.06 ± 0.14	31.13 ± 0.15	24.30 ± 0.56
NS (Grid)	6.19 ± 0.16	31.10 ± 0.18	25.21 ± 0.59
NS (Ring)	6.23 ± 0.15	30.85 ± 0.17	–
DST (Full)	13.07 ± 0.18	28.29 ± 0.17	22.19 ± 0.47
DST (Grid)	13.24 ± 0.18	28.36 ± 0.17	22.61 ± 0.29
DST (Ring)	13.24 ± 0.21	28.38 ± 0.17	–
DA (Full)	6.41 ± 0.17	27.48 ± 0.15	21.77 ± 0.35
DA (Grid)	6.44 ± 0.17	27.49 ± 0.19	21.82 ± 0.14
DA (Ring)	6.46 ± 0.18	27.57 ± 0.16	–
DOM-I (Full)	5.91 ± 0.12	27.27 ± 0.16	20.71 ± 0.37
DOM-I (Grid)	6.01 ± 0.13	27.34 ± 0.19	20.91 ± 0.46
DOM-I (Ring)	6.03 ± 0.13	27.37 ± 0.17	–
DOM-II (Full)	6.01 ± 0.12	27.68 ± 0.11	20.63 ± 0.34
DOM-II (Grid)	6.05 ± 0.08	27.71 ± 0.11	20.73 ± 0.32
DOM-II (Ring)	6.09 ± 0.12	27.72 ± 0.13	–

TABLE 5.1: Error Prediction Rate(%) and Standard Deviations(%) on Real-world Datasets

generalization performance compared with STL, and this property also holds under distributed settings if considering the good effect of the previous method (NS) by assigning one node to tackle one specific task. In the meantime, our new strategy of enabling each node to tackle all tasks successfully transforms STL algorithm (DA) into MTL methods. Among all the distributed methods, our proposed algorithm enjoys a lower error rate and it even outperforms traditional centralized learning on full-connected network, and is also competitive on grid-connected and ring-connected network.

Unlike our synthetic dataset, the relationship between real-world tasks is more complicated, where the optimal weight vector for each task may not be similar and outlier tasks often exist. The performance in Table 5.1 indicates: (1) Centralized OMTL algorithm (COML) still achieves better performance than learning each task individually. (2) The previous strategy of assigning each node to tackle one specific task (NS) actually increases the error prediction rate, and this can be easily explained by our previous analysis on ‘Pareto optimality’ where the optimal weight vectors may be quite different for

Figure 5.2: Above is the learning process on Sentiment dataset; below is the parameter effect on Synthetic and MHC-I dataset.

tasks. (3) Our strategy of information sharing still works well for both DA and DOM. By comparing with DST, which is actually a naive version of dispersing each task to all learning nodes, we can see the information sharing between tasks has a sound effect on distributed learning. Specifically, DOM algorithms show a favorable performance when compared with typical centralized OMTL algorithm, especially on networks with good connectivity properties. While the theoretical result suggests our algorithm achieves a $\mathcal{O}(\sqrt{T})$ regret bound while the counterpart in previous research (DA) is $\mathcal{O}(\sqrt{T} \log T)$, we observe the practical performance of DOM algorithms does outperform DA with lower error rates on all datasets in Table 5.1 and enjoys a faster learning progress in Figure 5.2.

Figure 5.2 also depicts the parameter effect for DOM with a constant and dynamic relationship matrix on full-connected networks¹. For algorithms with a constant B , we have given the theoretical optimal b_* in Theorem 5.7, and experiments validate our theoretical result as b_{exp} is rather close to b_* . We set $B_0 = I$ for DOM-II, and it's equal

¹Parameter effects are similar on grid and ring-connected network.

to learn each task independently if $\alpha = 1$. We can see a combination of an initial fixed relationship matrix and a dynamic term actually performs better than learning each task individually or setting the relationship to be totally dynamic.

Finally, it is important to note that the goal of this work is not to claim a new OMTL algorithm beating the existing centralized algorithms, but mainly focus on making OMTL algorithms feasible and effective under the challenging decentralized distributed settings. Experimental results show that the proposed decentralized algorithm **matches the performance** of the centralized OMTL learning (COML) on both synthetic and real-world datasets, clearly validating its efficacy.

5.5 Chapter Summary

In this chapter, we jump out of the stereotype of limiting a node as a ‘worker’ for a specific task and propose a novel learning framework so that these nodes are able to all tasks at the same time. The information-sharing scheme in our algorithms is separated into two sessions: multi-task’s information is first shared within each node and then the whole network is pushed towards a common minimizer by communication among different nodes. Such a learning framework not only works for our proposed algorithms, but also be applicable to transfer other STL algorithms (e.g., Dual Averaging) into distributed MTL algorithms. What’s more, we show that the proposed algorithm has $\mathcal{O}(\sqrt{T})$ regret upper bound if given a fixed relationship matrix, and later validate algorithms’ performance on both synthetic and real-world datasets.

5.6 Appendix: Proof for Lemmas

5.6.1 Proof for Eq (5.18)

By the definition of $\mathbf{z}_{t+1}$, we have

$$\begin{aligned}
 \mathbf{z}_{t+1} &= \frac{1}{m} \sum_{i=1}^m \mathbf{u}_{i,t+1} \\
 &= \frac{1}{m} \sum_{i=1}^m (\mathbf{v}_{i,t} + \mathbf{p}_{i,t+1}) \\
 &= \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^m a_{i,j} \mathbf{u}_{j,t} + \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t+1} \\
 &= \frac{1}{m} \sum_{j=1}^m \sum_{i=1}^m a_{i,j} \mathbf{u}_{j,t} + \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t+1} \\
 &= \frac{1}{m} \sum_{j=1}^m \mathbf{u}_{j,t} + \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t+1} \\
 &= \mathbf{z}_t + \frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t+1}
 \end{aligned}$$

Therefore, we have the following equation by recursion:

$$\mathbf{z}_{t+1} = \frac{1}{m} \sum_{i=1}^m \mathbf{u}_{i,0} + \frac{1}{m} \sum_{l=1}^{t+1} \sum_{i=1}^m \mathbf{p}_{i,l}$$

By assuming $\mathbf{u}_{i,0} = \mathbf{0}$, the above equation becomes

$$\mathbf{z}_{t+1} = \frac{1}{m} \sum_{l=1}^{t+1} \sum_{i=1}^m \mathbf{p}_{i,l}.$$

5.6.2 Proof for Lemma 5.4

Lemma 5.4. *Suppose the network is static and strongly connected, and assumptions 1,2,3 hold. We have the following inequality:*

$$\left| [\Phi(k, s)_{i,j} - \frac{1}{m}] \right| \leq \theta \beta^{k-s}$$

for all $k \geq s \geq 0$, where $\theta = (1 - \frac{\epsilon}{4m^2})^{-2}$ and $\beta = (1 - \frac{\epsilon}{4m^2})^{\frac{1}{Q}}$ only depend on the size and the topology of the network.

Proof: Proof for this lemma can be found in Lemma 3.2 [88]. □

5.6.3 Proof for Lemma 5.5

Lemma 5.5. Consider all the instances are within the range Ψ and Lemma 5.4 holds, then we have

$$\|\mathbf{z}_{t+1} - \mathbf{u}_{j,t+1}\|_{B_{\otimes}} \leq \eta L \Delta$$

where $\Delta = m\theta \frac{\beta}{1-\beta} + 2$.

Proof: By the definition in (5.17)(5.18), we have

$$\begin{aligned} & \|\mathbf{z}_{t+1} - \mathbf{u}_{j,t+1}\|_{B_{\otimes}} \\ &= \left\| \frac{1}{m} \sum_{l=1}^{t+1} \sum_{i=1}^m \mathbf{p}_{i,l} - \mathbf{p}_{j,t+1} - \sum_{l=1}^t \sum_{i=1}^m [\Phi(t+1, l)]_{j,i} \mathbf{p}_{i,l} \right\|_{B_{\otimes}} \\ &= \left\| \sum_{l=1}^t \sum_{i=1}^m \left(\frac{1}{m} - [\Phi(t+1, l)]_{j,i} \right) \mathbf{p}_{i,l} + \left(\frac{1}{m} \sum_{i=1}^m \mathbf{p}_{i,t+1} - \mathbf{p}_{j,t+1} \right) \right\|_{B_{\otimes}} \\ &\leq \sum_{l=1}^t \sum_{i=1}^m \left| \frac{1}{m} - [\Phi(t+1, l)]_{j,i} \right| \|\mathbf{p}_{i,l}\|_{B_{\otimes}} + \frac{1}{m} \sum_{i=1}^m \|\mathbf{p}_{i,t+1} - \mathbf{p}_{j,t+1}\|_{B_{\otimes}} \end{aligned}$$

The first term in above inequality can be written as:

$$\begin{aligned} & \sum_{l=1}^t \sum_{i=1}^m \left| \frac{1}{m} - [\Phi(t+1, l)]_{j,i} \right| \|\mathbf{p}_{i,l}\|_{B_{\otimes}} \\ &\stackrel{\text{Lemma 1}}{\leq} \sum_{l=1}^t \theta \beta^{t+1-l} \sum_{i=1}^m \|\mathbf{p}_{i,l}\|_{B_{\otimes}} \\ &\stackrel{(5.15)}{\leq} \sum_{l=1}^t \theta \beta^{t+1-l} m \eta L \\ &\leq m \theta \eta L \frac{\beta}{1-\beta} (1 - \beta^t) \end{aligned}$$

The second term can be bounded as,

$$\frac{1}{m} \sum_{i=1}^m \|\mathbf{p}_{i,t+1} - \mathbf{p}_{j,t+1}\|_{B_{\otimes}} \leq \frac{1}{m} \cdot m \cdot 2\eta L = 2\eta L.$$

Hence we have,

$$\begin{aligned} \|\mathbf{z}_{t+1} - \mathbf{u}_{j,t+1}\|_{B_{\otimes}} &\leq m\theta\eta L \frac{\beta}{1-\beta} (1 - \beta^t) + 2\eta L \\ &\leq \eta L (m\theta \frac{\beta}{1-\beta} + 2) \\ &= \eta L \Delta, \end{aligned}$$

where $\Delta = m\theta \frac{\beta}{1-\beta} + 2$. □

5.6.4 Proof for Lemma 5.6

Lemma 5.6. *The difference between $\mathbf{z}_t$ and any fixed $\mathbf{u}$ is bounded as*

$$\sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] \leq \frac{m\|\mathbf{u}\|_{B_{\otimes}}^2}{2\eta} + CL \sum_{t=1}^T \sum_{i=1}^m \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_{\otimes}} + \eta m T L^2 / 2.$$

Proof: For any $\mathbf{u}$, we have

$$\begin{aligned} \|\mathbf{u}_{i,t+1} - \mathbf{u}\|_{B_{\otimes}}^2 &= \|\mathbf{v}_{i,t} - \eta \nabla f(\mathbf{v}_{i,t}) - \mathbf{u}\|_{B_{\otimes}}^2 \\ &\leq \|\mathbf{v}_{i,t} - \mathbf{u}\|_{B_{\otimes}}^2 - 2\eta \langle \nabla f(\mathbf{v}_{i,t}), \mathbf{v}_{i,t} - \mathbf{u} \rangle + \|\eta \nabla f(\mathbf{v}_{i,t})\|_{B_{\otimes}}^2. \end{aligned}$$

By the convexity of f , we have

$$\langle \nabla f(\mathbf{v}_{i,t}), \mathbf{v}_{i,t} - \mathbf{u} \rangle \geq (f(\mathbf{v}_{i,t}) - f(\mathbf{u})),$$

and

$$\|\eta \nabla f(\mathbf{v}_{i,t})\|^2 \leq (\eta L)^2.$$

Therefore,

$$\|\mathbf{u}_{i,t+1} - \mathbf{u}\|_{B_\otimes}^2 \leq \|\mathbf{v}_{i,t} - \mathbf{u}\|_{B_\otimes}^2 - 2\eta[f(\mathbf{v}_{i,t}) - f(\mathbf{u})] + (\eta L)^2.$$

By the convexity of the squared norm, we have

$$\begin{aligned} \sum_{i=1}^m \|\mathbf{v}_{i,t+1} - \mathbf{u}\|_{B_\otimes}^2 &= \sum_{i=1}^m \left\| \sum_{j=1}^m a_{i,j} \mathbf{u}_{j,t+1} - \mathbf{u} \right\|_{B_\otimes}^2 \\ &\leq \sum_{i=1}^m \sum_{j=1}^m a_{i,j} \|\mathbf{u}_{j,t+1} - \mathbf{u}\|_{B_\otimes}^2 \\ &= \sum_{j=1}^m \|\mathbf{u}_{j,t+1} - \mathbf{u}\|_{B_\otimes}^2. \end{aligned}$$

So we have

$$\sum_{i=1}^m \|\mathbf{v}_{i,t+1} - \mathbf{u}\|_{B_\otimes}^2 \leq \sum_{i=1}^m \|\mathbf{v}_{i,t} - \mathbf{u}\|_{B_\otimes}^2 - 2\eta \sum_{i=1}^m [f(\mathbf{v}_{i,t}) - f(\mathbf{u})] + m(\eta L)^2.$$

On the other hand, we have

$$\begin{aligned} \sum_{i=1}^m [f(\mathbf{v}_{i,t}) - f(\mathbf{u})] &= \sum_{i=1}^m [f(\mathbf{v}_{i,t}) - f(\mathbf{z}_t)] + \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] \\ &\geq - \sum_{i=1}^m \|\nabla f\|_{B_\otimes^{-1}} \|\mathbf{z}_t - \mathbf{v}_{i,t}\|_{B_\otimes} + \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] \\ &= - \sum_{i=1}^m \|\nabla f\|_{B_\otimes^{-1}} \|\mathbf{z}_t - \sum_{j=1}^m a_{i,j} \mathbf{u}_{j,t}\|_{B_\otimes} + \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] \\ &\geq -CL \sum_{i=1}^m \|\mathbf{z}_t - \sum_{j=1}^m a_{i,j} \mathbf{u}_{j,t}\|_{B_\otimes} + \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] \\ &\geq -CL \sum_{i=1}^m \sum_{j=1}^m a_{i,j} \|\mathbf{z}_t - \mathbf{u}_{j,t}\|_{B_\otimes} + \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] \\ &\geq -CL \sum_{j=1}^m \|\mathbf{z}_t - \mathbf{u}_{j,t}\|_{B_\otimes} + \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})]. \end{aligned}$$

So we obtain

$$\begin{aligned} \sum_{i=1}^m \|\mathbf{v}_{i,t+1} - \mathbf{u}\|_{B_\otimes}^2 &\leq \sum_{i=1}^m \|\mathbf{v}_{i,t} - \mathbf{u}\|_{B_\otimes}^2 - 2\eta \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] \\ &\quad + 2\eta CL \sum_{i=1}^m \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_\otimes} + m(\eta L)^2. \end{aligned}$$

Summing it from 0 to $T - 1$,

$$\begin{aligned} &2\eta \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] \\ &\leq \sum_{t=1}^T \sum_{i=1}^m \|\mathbf{v}_{i,t} - \mathbf{u}\|_{B_\otimes}^2 - \sum_{t=1}^T \sum_{i=1}^m \|\mathbf{v}_{i,t+1} - \mathbf{u}\|_{B_\otimes}^2 + 2\eta CL \sum_{t=1}^T \sum_{i=1}^m \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_\otimes} + \sum_{t=1}^T m(\eta L)^2 \\ &= \sum_{i=1}^m \|\mathbf{v}_{i,0} - \mathbf{u}\|_{B_\otimes}^2 - \sum_{i=1}^m \|\mathbf{v}_{i,T} - \mathbf{u}\|_{B_\otimes}^2 + 2\eta CL \sum_{t=1}^T \sum_{i=1}^m \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_\otimes} + \sum_{t=1}^T m(\eta L)^2 \\ &\leq \sum_{i=1}^m \|\mathbf{v}_{i,0} - \mathbf{u}\|_{B_\otimes}^2 + 2\eta CL \sum_{t=1}^T \sum_{i=1}^m \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_\otimes} + \sum_{t=1}^T m(\eta L)^2 \\ &\leq m\|\mathbf{u}\|_{B_\otimes}^2 + 2\eta CL \sum_{t=1}^T \sum_{i=1}^m \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_\otimes} + mT\eta^2 L^2. \end{aligned}$$

Diving both sides by 2η , we comes to

$$\sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] \leq \frac{m\|\mathbf{u}\|_{B_\otimes}^2}{2\eta} + CL \sum_{t=1}^T \sum_{i=1}^m \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_\otimes} + \eta mTL^2/2.$$

□

5.6.5 Proof for Theorem 5.7

Theorem 5.7. Let $\mathbf{u}_*$ denotes the optimal classifier in hindsight, then the regret of the proposed DOM algorithm is upper bounded as

$$\sum_{t=1}^T \sum_{i=1}^m f(\mathbf{u}_{i,t}) - f(\mathbf{u}_*) \leq \frac{m\|\mathbf{u}_*\|_{B_\otimes}^2}{2\eta} + \eta mTL^2\left(\frac{1}{2} + 2C\Delta\right).$$

Proof: Based on Lemma 2, Lemma 3 and Eq (5.20), we have

$$\begin{aligned}
& \sum_{t=1}^T \sum_{i=1}^m f(\mathbf{u}_{i,t}) - f(\mathbf{u}_*) \\
& \stackrel{(5.20)}{\leq} \sum_{t=1}^T \sum_{i=1}^m [f(\mathbf{z}_t) - f(\mathbf{u})] + \sum_{t=1}^T \sum_{i=1}^m CL \|\mathbf{u}_{i,t} - \mathbf{z}_t\|_{B_\otimes} \\
& \stackrel{\text{Lemma 3}}{\leq} \frac{m \|\mathbf{u}_*\|_{B_\otimes}^2}{2\eta} + 2CL \sum_{t=1}^T \sum_{i=1}^m \|\mathbf{z}_t - \mathbf{u}_{i,t}\|_{B_\otimes} + \eta m T L^2 / 2 \\
& \stackrel{\text{Lemma 2}}{\leq} \frac{m \|\mathbf{u}_*\|_{B_\otimes}^2}{2\eta} + 2\eta m T C L^2 \Delta + \eta m T L^2 / 2 \\
& = \frac{m \|\mathbf{u}_*\|_{B_\otimes}^2}{2\eta} + \eta m T L^2 \left(\frac{1}{2} + 2C\Delta \right)
\end{aligned}$$

With $\eta = \frac{\|\mathbf{u}_*\|_{B_\otimes}}{L\sqrt{T(1+4C\Delta)}}$, we obtain $R(T) = mL\sqrt{(T+4C\Delta)\|\mathbf{u}_*\|_{B_\otimes}^2} = \mathcal{O}(\sqrt{T})$

□

Chapter 6

Conclusion and Future Work

6.1 Conclusion

Online federated learning on decentralized networks plays an important role in large-scale machine learning, especially when datasets are distributed over multiple nodes or instances cannot be exchanged. In the previous chapters, we have investigated and solved several open challenges in decentralized optimization: one is to reduce the communication burden for dense networks and improve the communication efficiency; another is extending the current accuracy-based estimation to AUC-oriented optimization and the general convex-concave optimization; the third is extending the single-task to multi-task learning.

For the first challenge, we adopt a novel communication protocol based on edge sampling and allow each node to sample a subset of its neighbors to communicate rather than the whole neighborhood, which actually balances the computation burden of internal computation and external communication. Different from previous biased sampling strategy, the new sampling method is known to be strictly unbiased and also suffers from less variance. Most importantly, this sampling algorithm is shown to be highly communication-efficient and enjoys faster a convergence rate than unsampled algorithm with the same communication budget, which is validated with both theoretical analysis and experimental results.

We also notice that most of the current works focus on unconstrained problems and target at optimizing accuracy, and therefore present an online federated convex-concave algorithm. The proposed algorithm provides a solution for deploying a series of challenges on decentralized networks, such as primal-dual methods in optimization theory and online AUC maximization in machine learning. With clear $\mathcal{O}(\sqrt{T})$ upper and lower regret bounds, we theoretically guarantee the performance of our proposed algorithm, as well as revealing its relationship with the network size and topology. This theoretical result is further confirmed by real-world experiments for online AUC maximization, which turns out to be a byproduct of our general solution but stands out as the first decentralized online AUC maximization algorithm.

Finally, we investigate a more complicated scenario where multiple tasks are distributed over the networks and nodes on the graph need to collaborate to optimize this multi-task problem. We present the challenge of ‘Pareto optimality’ when dealing with multi-task optimization and jump out the stereotype of considering each node as a worker for a special task. Instead, we decompose the communication into two phases: internal communication for multi-task information sharing and external communication for reaching consensus. Such a method is shown to both guarantee boosting the performance of each single task and reaching a global optimum for the whole network.

6.2 Future Work

In this thesis, we have proposed solutions to a number of challenging and practical problems in the area of decentralized optimizations, but there are still some potential issues awaiting our future exploration.

6.2.1 Sufficient Updating Local Model

The current online federated learning iteratively adopts internal adaptation and external communication, and often assume the cost of these two procedures are similar. In the previous chapter, we have considered reducing the communication cost by adopting edge sampling, but still didn’t jump out of this ‘combine-and-the-adapt’ framework.

In many cases, the frequent communication may be impossible and it's preferred to build an effective local model before communication. For centralized distributed learning, such a problem has been efficiently solved by several algorithms like CoCoA [89, 90] and DADM [91] where the authors consider the dual formulation and update the local model to certain precision before communication.

However, in these algorithms, the 'master' node acts as an important coordinator to decide when to communicate and how to combine local information. It still remains unknown on how to remove the coordinator and simply use diffusion to transport information. A possible issue of the new framework is how to avoid local overfitting, as we have shown in Figure 3.2 that without sufficient communication synchronous gossip algorithm actually suffers from the worse performance.

6.2.2 Non-convex Optimization

Another potential future work is the extension for non-convex cases. One recent work in [92] shows the traditional distributed gradient descent (DGD), as well as its proximal variants, still retain most properties that are known in the convex setting and is able to reach a consensus stationary solution with a speed of $\mathcal{O}(\frac{1}{k^\epsilon})$ for a certain $\epsilon \in (0, 1]$.

But we notice this work simply inherits the general framework designed for convex optimization and misses a few special characteristics of non-convex optimization. The first the over-parameterization problem in the field of deep learning. Unlike convex optimization where we often assume the instances are significantly more than the feature parameters, in modern deep learning, the over-parameterization problem is quite common and leads to new challenges when distributing datasets over the networks. In this case, the point-to-point convex feasibility problem (CFP) [93] turns out to be seeking an intersection of (non)convex sets. Reaching a stationary point is far from our satisfactory since most of these points are out of the intersection. We expect to design distributed algorithms to seek for an effective solution.

In the meanwhile, previous studies demonstrate this over-parameterization on networks may have certain effects in the training, such as implicit acceleration [94] and

regularization [95]. Whether such properties remain in distributed settings remains unknown and worth our future exploration.

Appendix A

List of Publications

1. Chi Zhang, Peilin Zhao, Shuji Hao, Yeng Chai Soh, and Bu Sung Lee. Rom: A robust online multi-task learning approach. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1341–1346. IEEE, 2016
2. Shuji Hao, Peilin Zhao, Jing Lu, Steven CH Hoi, Chunyan Miao, and Chi Zhang. Soal: Second-order online active learning. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 931–936. IEEE, 2016
3. Shuji Hao, Jing Lu, Peilin Zhao, Chi Zhang, Steven CH Hoi, and Chunyan Miao. Second-order online active learning and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 2017
4. Chi Zhang, Peilin Zhao, Shuji Hao, Yeng Chai Soh, Bu Sung Lee, Chunyan Miao, and Steven CH Hoi. Distributed multi-task classification: a decentralized online learning approach. *Machine Learning*, 107(4):727–747, 2018

Bibliography

- [1] Joel B Predd, Sanjeev R Kulkarni, and H Vincent Poor. Distributed kernel regression: An algorithm for training collaboratively. In *Information Theory Workshop, 2006. ITW'06 Punta del Este. IEEE*, pages 332–336. IEEE, 2006.
- [2] Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *OSDI*, volume 14, pages 583–598, 2014.
- [3] Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. In *Advances in Neural Information Processing Systems*, pages 19–27, 2014.
- [4] J-P Goux, Sanjeev Kulkarni, Jeff Lindereth, and Michael Yoder. An enabling framework for master-worker applications on the computational grid. In *High-Performance Distributed Computing, 2000. Proceedings. The Ninth International Symposium on*, pages 43–50. IEEE, 2000.
- [5] Philip E Agre and Marc Rotenberg. *Technology and privacy: The new landscape*. Mit Press, 1998.
- [6] Jakub Konecny, Brendan McMahan, and Daniel Ramage. Federated optimization: Distributed optimization beyond the datacenter. *arXiv preprint arXiv:1511.03575*, 2015.
- [7] Michael Rabbat and Robert Nowak. Distributed optimization in sensor networks. In *Proceedings of the 3rd international symposium on Information processing in sensor networks*, pages 20–27. ACM, 2004.

- [8] Ion Necoara. Random coordinate descent algorithms for multi-agent convex optimization over networks. *IEEE Transactions on Automatic Control*, 58(8):2001–2012, 2013.
- [9] Yongcan Cao, Wenwu Yu, Wei Ren, and Guanrong Chen. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Transactions on Industrial informatics*, 9(1):427–438, 2013.
- [10] Reza Olfati-Saber and Nils F Sandell. Distributed tracking in sensor networks with limited sensing range. In *American Control Conference, 2008*, pages 3157–3162. IEEE, 2008.
- [11] Li Chunlin and Li Layuan. A distributed multiple dimensional qos constrained resource scheduling optimization policy in computational grid. *Journal of Computer and System Sciences*, 72(4):706–726, 2006.
- [12] Ali H Sayed et al. Adaptation, learning, and optimization over networks. *Foundations and Trends® in Machine Learning*, 7(4-5):311–801, 2014.
- [13] Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.
- [14] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.
- [15] Albert BJ Novikoff. On convergence proofs for perceptrons. Technical report, DTIC Document, 1963.
- [16] Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, 2006.
- [17] Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. 2003.
- [18] Elad Hazan, Amit Agarwal, and Satyen Kale. Logarithmic regret algorithms for online convex optimization. *Machine Learning*, 69(2-3):169–192, 2007.

- [19] Hal Daumé III and Daniel Marcu. Learning as search optimization: Approximate large margin methods for structured prediction. In *Proceedings of the 22nd international conference on Machine learning*, pages 169–176. ACM, 2005.
- [20] Yi Li and Philip M Long. The relaxed online maximum margin algorithm. In *Advances in neural information processing systems*, pages 498–504, 2000.
- [21] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of Computing*, 8(1):121–164, 2012.
- [22] Brendan McMahan. Follow-the-regularized-leader and mirror descent: Equivalence theorems and ℓ_1 regularization. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 525–533, 2011.
- [23] Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. A second-order perceptron algorithm. *SIAM Journal on Computing*, 34(3):640–668, 2005.
- [24] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM, 2008.
- [25] Koby Crammer, Alex Kulesza, and Mark Dredze. Adaptive regularization of weight vectors. In *Advances in neural information processing systems*, pages 414–422, 2009.
- [26] Jialei Wang, Peilin Zhao, and Steven CH Hoi. Exact soft confidence-weighted learning. *arXiv preprint arXiv:1206.4612*, 2012.
- [27] Haipeng Luo, Alekh Agarwal, Nicolo Cesa-Bianchi, and John Langford. Efficient second order online learning by sketching. In *Advances in Neural Information Processing Systems*, pages 902–910, 2016.
- [28] Steven CH Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. Online learning: A comprehensive survey. *arXiv preprint arXiv:1802.02871*, 2018.
- [29] Steven CH Hoi, Jialei Wang, and Peilin Zhao. Libol: A library for online learning algorithms. *The Journal of Machine Learning Research*, 15(1):495–499, 2014.

- [30] John Nikolas Tsitsiklis. Problems in decentralized decision making and computation. Technical report, DTIC Document, 1984.
- [31] John Tsitsiklis, Dimitri Bertsekas, and Michael Athans. Distributed asynchronous deterministic and stochastic gradient optimization algorithms. *IEEE transactions on automatic control*, 31(9):803–812, 1986.
- [32] Joel B Predd, Sanjeev R Kulkarni, and Harold Vincent Poor. Consistency in models for distributed learning under communication constraints. *IEEE Transactions on Information Theory*, 52(1):52–63, 2006.
- [33] Kun Yuan, Qing Ling, and Wotao Yin. On the convergence of decentralized gradient descent. *SIAM Journal on Optimization*, 26(3):1835–1854, 2016.
- [34] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. Extra: An exact first-order algorithm for decentralized consensus optimization. *SIAM Journal on Optimization*, 25(2):944–966, 2015.
- [35] Wei Shi, Qing Ling, Gang Wu, and Wotao Yin. A proximal gradient algorithm for decentralized composite optimization. *IEEE Transactions on Signal Processing*, 63(22):6013–6023, 2015.
- [36] Aryan Mokhtari and Alejandro Ribeiro. Dsa: Decentralized double stochastic averaging gradient algorithm. *The Journal of Machine Learning Research*, 17(1): 2165–2199, 2016.
- [37] John C Duchi, Alekh Agarwal, and Martin J Wainwright. Dual averaging for distributed optimization: convergence analysis and network scaling. *IEEE Transactions on Automatic Control*, 57(3):592–606, 2012.
- [38] Wei Shi, Qing Ling, Kun Yuan, Gang Wu, and Wotao Yin. On the linear convergence of the admm in decentralized consensus optimization. *IEEE Trans. Signal Processing*, 62(7):1750–1761, 2014.
- [39] Kevin Scaman, Francis Bach, Sébastien Bubeck, Yin Tat Lee, and Laurent Massoulié. Optimal algorithms for smooth and strongly convex distributed optimization in networks. *arXiv preprint arXiv:1702.08704*, 2017.

- [40] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Convex multi-task feature learning. *Machine Learning*, 73(3):243–272, 2008.
- [41] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *The Journal of Machine Learning Research*, 6:1817–1853, 2005.
- [42] Andreas Maurer. Bounds for linear multi-task learning. *Journal of Machine Learning Research*, 7(Jan):117–139, 2006.
- [43] Jacob Abernethy, Peter Bartlett, and Alexander Rakhlin. Multitask learning with expert advice. In *Learning Theory*, pages 484–498. Springer, 2007.
- [44] Gábor Lugosi, Omiros Papaspiliopoulos, and Gilles Stoltz. Online multi-task learning with hard constraints. *arXiv preprint arXiv:0902.3526*, 2009.
- [45] Giovanni Cavallanti, Nicolo Cesa-Bianchi, and Claudio Gentile. Linear algorithms for online multitask classification. *The Journal of Machine Learning Research*, 11:2901–2934, 2010.
- [46] Avishek Saha, Piyush Rai, Suresh Venkatasubramanian, and Hal Daume. Online learning of multiple tasks and their relationships. In *International Conference on Artificial Intelligence and Statistics*, pages 643–651, 2011.
- [47] Yu Zhang and Qiang Yang. A survey on multi-task learning. *arXiv preprint arXiv:1707.08114*, 2017.
- [48] Stephen Boyd, Persi Diaconis, and Lin Xiao. Fastest mixing markov chain on a graph. *SIAM review*, 46(4):667–689, 2004.
- [49] David A Levin and Yuval Peres. *Markov chains and mixing times*, volume 107. American Mathematical Soc., 2017.
- [50] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Randomized gossip algorithms. *IEEE transactions on information theory*, 52(6):2508–2530, 2006.

- [51] S Sitharama Iyengar, Richard R Brooks, et al. *Distributed sensor networks*. CRC press, 2004.
- [52] Maria Florina Balcan, Avrim Blum, Shai Fine, and Yishay Mansour. Distributed learning, communication complexity and privacy. In *Conference on Learning Theory*, pages 26–1, 2012.
- [53] David P Woodruff and Qin Zhang. When distributed computation is communication expensive. *Distributed Computing*, 30(5):309–323, 2017.
- [54] S Sundhar Ram, A Nedić, and Venugopal V Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications*, 147(3):516–545, 2010.
- [55] Yuchen Zhang, John Duchi, Michael I Jordan, and Martin J Wainwright. Information-theoretic lower bounds for distributed statistical estimation with communication constraints. In *Advances in Neural Information Processing Systems*, pages 2328–2336, 2013.
- [56] Paul Erdős and Alfréd Rényi. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(17-61):43, 1960.
- [57] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.
- [58] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.
- [59] Angelia Nedić and Asuman Ozdaglar. Subgradient methods for saddle-point problems. *Journal of optimization theory and applications*, 142(1):205–228, 2009.
- [60] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic game theory*, volume 1. Cambridge University Press Cambridge, 2007.
- [61] Yuchen Zhang and Xiao Lin. Stochastic primal-dual coordinate method for regularized empirical risk minimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 353–361, 2015.

- [62] Yiming Ying, Longyin Wen, and Siwei Lyu. Stochastic online auc maximization. In *Advances in Neural Information Processing Systems*, pages 451–459, 2016.
- [63] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.
- [64] Peilin Zhao, Rong Jin, Tianbao Yang, and Steven C Hoi. Online auc maximization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 233–240, 2011.
- [65] Corinna Cortes and Mehryar Mohri. Auc optimization vs. error rate minimization. *Advances in neural information processing systems*, 16(16):313–320, 2004.
- [66] Wei Gao, Rong Jin, Shenghuo Zhu, and Zhi-Hua Zhou. One-pass auc optimization. *arXiv preprint arXiv:1305.1363*, 2013.
- [67] Wei Gao and Zhi-Hua Zhou. On the consistency of auc pairwise optimization. In *IJCAI*, pages 939–945, 2015.
- [68] Duncan J Watts and Steven H Strogatz. Collective dynamics of ‘small-world’ networks. *nature*, 393(6684):440, 1998.
- [69] Wenpeng Zhang, Peilin Zhao, Wenwu Zhu, Steven CH Hoi, and Tong Zhang. Projection-free distributed online learning in networks. In *International Conference on Machine Learning*, pages 4054–4062, 2017.
- [70] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.
- [71] Sahand Negahban and Martin J Wainwright. Joint support recovery under high-dimensional scaling: Benefits and perils of l1-regularization. *Advances in Neural Information Processing Systems*, 21:1161–1168, 2008.
- [72] Ting Kei Pong, Paul Tseng, Shuiwang Ji, and Jieping Ye. Trace norm regularization: Reformulations, algorithms, and multi-task learning. *SIAM Journal on Optimization*, 20(6):3465–3489, 2010.
- [73] Ofer Dekel, Philip M Long, and Yoram Singer. Online multitask learning. In *Learning Theory*, pages 453–467. Springer, 2006.

- [74] Guangxia Li, Steven CH Hoi, Kuiyu Chang, Wenting Liu, and Ramesh Jain. Collaborative online multitask learning. *Knowledge and Data Engineering, IEEE Transactions on*, 26(8):1866–1876, 2014.
- [75] Chi Zhang, Peilin Zhao, Shuji Hao, Yeng Chai Soh, and Bu Sung Lee. Rom: A robust online multi-task learning approach. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 1341–1346. IEEE, 2016.
- [76] Francesco Dinuzzo, Gianluigi Pillonetto, and Giuseppe De Nicolao. Client–server multitask learning from distributed datasets. *IEEE Transactions on Neural Networks*, 22(2):290–303, 2011.
- [77] Jialei Wang, Mladen Kolar, and Nathan Srebro. Distributed multi-task learning with shared representation. *arXiv preprint arXiv:1603.02185*, 2016.
- [78] Jialei Wang, Mladen Kolar, Nathan Srebro, et al. Distributed multi-task learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 751–760, 2016.
- [79] Jie Chen, Cédric Richard, and Ali H Sayed. Multitask diffusion adaptation over networks. *IEEE Transactions on Signal Processing*, 62(16):4129–4144, 2014.
- [80] Alexander Bertrand and Marc Moonen. Distributed adaptive node-specific signal estimation in fully connected sensor networks—part i: Sequential node updating. *IEEE Transactions on Signal Processing*, 58(10):5277–5291, 2010.
- [81] Alexander Bertrand and Marc Moonen. Distributed adaptive estimation of node-specific signals in wireless sensor networks with a tree topology. *IEEE Transactions on Signal Processing*, 59(5):2196–2210, 2011.
- [82] Theodoros Evgeniou, Charles A Micchelli, and Massimiliano Pontil. Learning multiple tasks with kernel methods. In *Journal of Machine Learning Research*, pages 615–637, 2005.
- [83] Keerthiram Murugesan, Hanxiao Liu, Jaime Carbonell, and Yiming Yang. Adaptive smoothed online multi-task learning. In *Advances in Neural Information Processing Systems*, pages 4296–4304, 2016.

- [84] Sheng-Yuan Tu and Ali H Sayed. Diffusion strategies outperform consensus strategies for distributed estimation over adaptive networks. *IEEE Transactions on Signal Processing*, 60(12):6217–6234, 2012.
- [85] Gilbert Strang, Gilbert Strang, Gilbert Strang, and Gilbert Strang. *Introduction to linear algebra*, volume 3. Wellesley-Cambridge Press Wellesley, MA, 1993.
- [86] Bjoern Peters, Huynh-Hoa Bui, Sune Frankild, Morten Nielsen, Claus Lundegaard, Emrah Kostem, Derek Basch, Kasper Lamberth, Mikkel Harndahl, Ward Fleri, et al. A community resource benchmarking predictions of peptide binding to mhc-i molecules. *PLoS Comput Biol*, 2(6):e65, 2006.
- [87] John Blitzer, Mark Dredze, Fernando Pereira, et al. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *ACL*, volume 7, pages 440–447, 2007.
- [88] S Sundhar Ram, Angelia Nedić, and Venugopal V Veeravalli. Distributed stochastic subgradient projection algorithms for convex optimization. *Journal of optimization theory and applications*, 147(3):516–545, 2010.
- [89] Martin Jaggi, Virginia Smith, Martin Takác, Jonathan Terhorst, Sanjay Krishnan, Thomas Hofmann, and Michael I Jordan. Communication-efficient distributed dual coordinate ascent. In *Advances in neural information processing systems*, pages 3068–3076, 2014.
- [90] Virginia Smith, Simone Forte, Chenxin Ma, Martin Takac, Michael I Jordan, and Martin Jaggi. Cocoa: A general framework for communication-efficient distributed optimization. *arXiv preprint arXiv:1611.02189*, 2016.
- [91] Shun Zheng, Jiale Wang, Fen Xia, Wei Xu, and Tong Zhang. A general distributed dual coordinate optimization framework for regularized loss minimization. *The Journal of Machine Learning Research*, 18(1):4096–4117, 2017.
- [92] Jinshan Zeng and Wotao Yin. On nonconvex decentralized gradient descent. *IEEE Transactions on Signal Processing*, 66(11):2834–2848, 2018.

- [93] Heinz H Bauschke and Jonathan M Borwein. On projection algorithms for solving convex feasibility problems. *SIAM review*, 38(3):367–426, 1996.
- [94] Sanjeev Arora, Nadav Cohen, and Elad Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. *arXiv preprint arXiv:1802.06509*, 2018.
- [95] Yuanzhi Li, Tengyu Ma, and Hongyang Zhang. Algorithmic regularization in overparameterized matrix sensing and neural networks with quadratic activations. In *Conference On Learning Theory*, pages 2–47, 2018.
- [96] Shuji Hao, Peilin Zhao, Jing Lu, Steven CH Hoi, Chunyan Miao, and Chi Zhang. Soal: Second-order online active learning. In *Data Mining (ICDM), 2016 IEEE 16th International Conference on*, pages 931–936. IEEE, 2016.
- [97] Shuji Hao, Jing Lu, Peilin Zhao, Chi Zhang, Steven CH Hoi, and Chunyan Miao. Second-order online active learning and its applications. *IEEE Transactions on Knowledge and Data Engineering*, 2017.
- [98] Chi Zhang, Peilin Zhao, Shuji Hao, Yeng Chai Soh, Bu Sung Lee, Chunyan Miao, and Steven CH Hoi. Distributed multi-task classification: a decentralized online learning approach. *Machine Learning*, 107(4):727–747, 2018.