COMMUNICATION-EFFICIENT OPTIMIZATION
AND LEARNING FOR DISTRIBUTED MULTI-AGENT SYSTEMS

by

Ping Xu
A Dissertation
Submitted to the
Graduate Faculty
of
George Mason University
In Partial fulfillment of
The Requirements for the Degree
of
Doctor of Philosophy
Electrical Engineering

Committee:

_____ Dr. Zhi Tian, Dissertation Director

_____ Dr. Yariv Ephraim, Committee Member

_____ Dr. Kathleen E. Wage, Committee Member

_____ Dr. Kuo-Chu Chang, Committee Member

_____ Dr. Monson H. Hayes, Department Chair

Date: _____ Spring 2022
George Mason University
Fairfax, VA

Communication-Efficient Optimization and Learning for Distributed Multi-agent Systems

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy at George Mason University

By

Ping Xu
Master of Science
George Mason University, May 2018
Bachelor of Science
Northwestern Polytechnical University, July 2015

Director: Dr. Zhi Tian, Professor
Department of Electrical and Computer Engineering

Spring 2022
George Mason University
Fairfax, VA

Dedication

This dissertation is dedicated to my family and friends for their unconditional love and support.

Acknowledgments

First and foremost, I would like to thank my Ph.D. advisor Prof. Zhi Tian for her guidance and constant encouragement, which makes me become not only a better researcher, but also a better person. This thesis would not have been possible without her insightful suggestions. I would like to extend my appreciation to Professors Yariv Ephraim, Kathleen E. Wage, and Kuo-Chu Chang who agreed to take time out of their busy schedules and serve on my doctoral committee. The materials in this thesis also benefited from discussions with my collaborators Dr. Yue Wang and Dr. Yaohua Liu. I am truly grateful for their continuous help. Finally, I want to thank my parents for their love, support, and belief in me.

Table of Contents

List of Tables

List of Figures

Abstract

COMMUNICATION-EFFICIENT OPTIMIZATION AND LEARNING FOR DISTRIBUTED MULTI-AGENT SYSTEMS

Ping Xu, PhD

George Mason University, 2022

Dissertation Director: Dr. Zhi Tian

Distributed learning has attracted extensive interest in recent years, owing to the explosion of data generated from mobile sensors, social media services, and other networked multi-agent applications. In many of these applications, the observed data are usually kept private at local sites without being aggregated to a fusion center, either due to the prohibitively high cost of raw data transmission or privacy concerns. Meanwhile, each agent in the network only communicates with its neighbors within a one-hop local range to save transmission power. Moreover, distributed learning is typically implemented in an iterative manner for computational feasibility and efficiency. This incurs frequent communications among agents to exchange their locally computed updates of the shared learning model, which can cause tremendous communication overhead in terms of both link bandwidth and transmission power. Under this circumstance, this dissertation focuses on developing communication-efficient distributed learning algorithms for multi-agent systems under communication and privacy constraints.

To be specific, we utilize the random feature (RF) mapping method to circumvent the curse of dimensionality issue in traditional kernel methods and bypass transmitting raw data in distributed kernel learning. This approach enables the reformulation of decentralized

kernel learning as a decentralized consensus optimization problem in the RF space, which is then solved distributedly and iteratively via the alternating direction method of multipliers (ADMM), with a communication-censoring strategy incorporated to evaluate if an update is informative enough to be transmitted. For online streaming data with possibly unknown dynamics, this dissertation develops corresponding adaptive and dynamic decentralized learning approaches to learn the optimal function "on the fly" via linearized ADMM and conventional decentralized ADMM, respectively. Communication censoring and quantization strategies are utilized for both approaches to save communication resources. Finally, the present dissertation offers a unified framework of learning nonlinear input-output maps by bridging the gap in kernel methods and neural networks, which leads to the development of an RF-based deep kernel learning network with multiple learning paths.

Chapter 1: Introduction

1.1 Motivation and context

Distributed multi-agent systems are deployed in broad applications such as unmanned vehicles, distributed sensor networks, social media, health care, and environmental monitoring devices, to name a few [1–5]. Here, the term *agent* [1] can be a single computational device (e.g. smart phone, database, wireless sensor, etc.) or a collection of co-located computational systems (e.g. data centers, computer clusters, etc.). The explosion of data generated from these systems, as well as technological advances in data collection and storage, have propelled the ongoing boom of machine learning (ML) technologies which aim to automatically learn useful information from data in order to attain high-performing predictive models to assist future decision making.

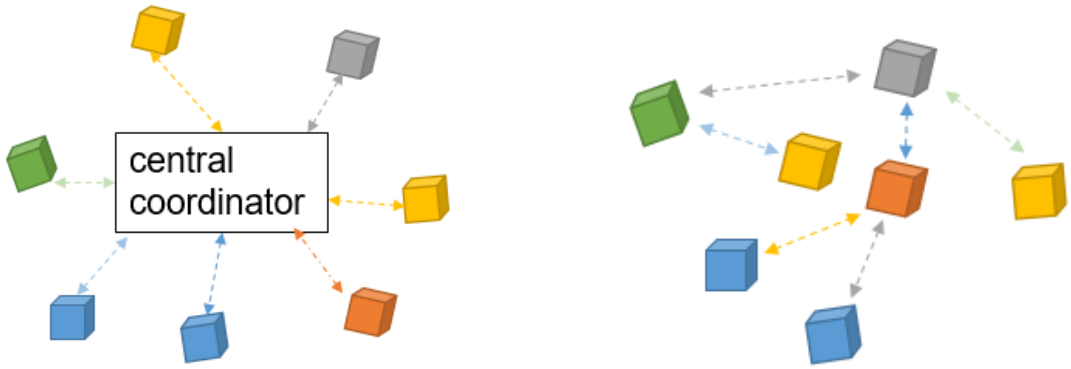
ML is usually intertwined with mathematical optimization in the sense that an ML model is often trained by solving an optimization problem whose objective function is data-dependent. Traditional optimization approaches for ML work in a centralized way, in which all training data are moved to a central node for further processing. While such centralized approaches can attain high accuracy, their implementation in many aforementioned applications is unviable. This is because the observed data in many multi-agent systems are kept private at local sites without being aggregated to a fusion center, due to either the prohibitively high cost of raw data transmission or privacy and security concerns during transmissions and centralized storage. Meanwhile, data collection is increasingly done by massive sensing devices or distributed data centers where each agent in the network only communicates with its neighbors within the one-hop local communication range to save

[1] Throughout this dissertation, "agent" and "node" are interchangeably used since the communication network of the multi-agent system is usually described by graphs whose elements are nodes.

transmission power. As a result, distributed learning is well-motivated, which has attracted extensive interest in recent years in both academia and industry.

To solve a learning problem in a distributed manner, a number of distributed agents collaboratively carry out a common learning task based on a shared ML model, without exchanging their local private raw data. Global learning is then realized through network-level optimization involving both local computation and communications among agents, where the latter is especially important when the local data is inadequate to accurately learn the ML model. Depending on the communication network topology, two distributed learning paradigms arise (Figure 1.1): conventional federated learning in which all nodes communicate to a central coordinator through a star topology [6, 7], and fully decentralized (federated) learning in which all agents exchange information only with their neighbors within a one-hop local communication range, without central coordination [8,9]. Under both paradigms, distributed learning algorithms are typically implemented by iteratively computing the model updates for computational feasibility and efficiency, and for coping with streaming data as well. This incurs frequent communications among agents to exchange their locally computed updates of the shared learning model, which can cause tremendous communication overhead in terms of both link bandwidth and transmission power. Hence, the communication efficiency in implementing iterative algorithms is of utmost importance, which underpins the practical implementation of all distributed learning techniques, including (semi)-supervised, unsupervised, and reinforcement learning.

For federated learning that adopts a star topology, the central node plays a crucial role in not only synchronizing the iterative updating algorithms running at distributed nodes during parallel computing but also performing possibly centralized communication resource allocation [10]. However, for a fully decentralized network, each node has to make autonomous decisions on the global learning task via communicating with linked or adjacent nodes only, which is much more challenging. Nevertheless, due to the hindrance of node synchronization requirements in parallel computing, there is growing interest in



(a) Centralized topology distributed data.

(b) Decentralzied topology distributed data.

Figure 1.1: Distributed learning paradigms: (a) conventional federated learning; (b) decentralized learning with no central coordination.

fully decentralized learning to take advantage of its built-in robustness to both node failure and asynchronous computing in heterogeneous environments, at no cost of convergence rates compares with its synchronous centralized counterpart [11, 12]. This dissertation thus focuses on investigating learning over decentralized networks (Figure 1.1(b)) since it is technically more challenging and much less studied so far given its unique technical issues related to big data analytics. Nonetheless, the algorithms we develop can be easily applied to federated learning with centralized topology.

Conforming to the emerging need in data science, this dissertation aims to develop efficient decentralized optimization techniques for distributed learning. Among many successful centralized ML techniques, kernel methods and multilayer neural networks (NNs) are representatives of the nonparametric learning and parametric methods, respectively. They are both attractive for various learning tasks such as regression, classification, clustering, as well as reinforcement learning [13, 14], owing to their abilities in modeling the (complex) nonlinear relationships behind the input-output samples. However, kernel methods adopt nonparametric models where the number of model variables grows proportionally to the data size. Under such a curse of dimensionality, it is formidably challenging to implement kernel methods in a distributed or adaptive fashion when the data sizes vary at different

locations and across time. On the other hand, despite the practical success of NNs, they rely much on intuition, heuristics, and trial-and-error, whereas the theoretical understanding of them is still lagging. Thus, research on the distributed implementation of these methods with a theoretical foundation is still at its early stage, which calls for substantial efforts and investment to clearly understand its niche role in big data and real-time streaming applications, and to propel its development toward maturity. Moreover, an overwhelming amount of data is transmitted nowadays over wireless networks to power intelligent applications, making communication efficiency a priority, not an afterthought. To this end, this dissertation strives to systematically address the following challenging questions on distributed learning:

- Communication-efficient learning: how can we design distributed optimization techniques with built-in communication efficiency and provable convergence for practical ML scenarios involving non-parametric kernel models in decentralized networks?
- Online adaptive learning: how can we enhance the adaptivity and computation efficiency of communication-efficient distributed kernel learning for streaming data?
- Online dynamic learning: how can we perform distributed kernel learning under environments with unknown dynamics in a communication-efficient manner?
- Bridging kernel methods and neural networks: what are the differences/connections between kernel methods and multilayer neural networks, and can we develop a method that possesses the strengths of both methods and overcomes their weaknesses?

The key outcomes of this dissertation are algorithms and analysis of distributed ML tools for big data analytics, including scalable nonlinear learning from high-dimensional networked data. Such results may find broad applications in both designing data-driven distributed social, biological, and financial systems and understanding their structures and dynamics.

1.2 Network and communication models

Throughout this dissertation, we consider the decentralized network whose underlying graph is in Figure 1.1 (b). The communication scheme considered is synchronous, although the communication-efficient techniques employed in this dissertation can also be applied to asynchronous scenarios. Next, we give a detailed description of the network and communication models.

Network Model. Consider a bidirectionally connected network of N agents and r arcs, whose underlying undirected communication graph is denoted as $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, where $\mathcal{N}$ is the set of agents with cardinality $|\mathcal{N}| = N$ and $\mathcal{A}$ is the set of undirected arcs with cardinality $|\mathcal{A}| = 2r$. Two agents i and j are called as neighbors when $(i, j) \in \mathcal{A}$ and, by the symmetry of the network, $(j, i) \in \mathcal{A}$. For agent i, its neighbors within one-hop local communication range neighbors are in the set $\mathcal{N}_i = \{j|(j, i) \in \mathcal{A}\}$. The cardinality $|\mathcal{N}_i|$ is also known as the degree d_i of agent i. The degree matrix of the communication graph is $\mathbf{D} \in \mathbb{R}^{N\times N}$ which is diagonal with the ith diagonal element being $d_i, \forall i$, and the symmetric adjacency matrix associated with the communication graph is $\mathbf{W} \in \mathbb{R}^{N\times N}$, whose (i, j)th entry is 1 if agent i and j are neighbors or 0 otherwise.

Communication Model. In this dissertation, we consider synchronous communication during ML training. For batch-form learning where agents have all the training data, the iterative process of algorithm implementation consists of two alternating stages per iteration: communication and computation. In the communication stage, each agent broadcasts its state variable to its neighbors and receives state variables from its neighbors according to the communication censoring rule which shall be introduced later. In the computation stage, each agent carries out local updates based on its local objective function and state variables. This two-stage iterative process continues until the algorithm converges or meets the desired result. For online learning from streaming data, an observation stage is conducted between the communication and computation stages. That is, after communicating

with its neighbors, each agent collects its streaming data and formulates its local objective function in the observation stage. Then, in the computation stage, each agent carries out local updates based on the local objective function, state variables, and the observed data.

Under such network and communication models, decentralized learning amounts to designing both the local computation rule and communication policy, to reach the globally optimal estimate of the shared learning model with provable convergence, in the absence of any centralized aggregation or coordination.

1.3 Thesis outline and published results

This dissertation deals with communication-efficient scalable and adaptive distributed learning for networked multi-agent systems, along with some exploratory efforts to bridge the nonparametric kernel methods and parametric neural networks. The outline of this dissertation is presented next, followed by Table 1.1 that summarizes the main results and relevant publications.

1.3.1 Communication-censored decentralized kernel learning

This dissertation starts with a general setting of decentralized kernel learning in complex yet static environments. Nonparametric kernel methods are widely utilized in many nonlinear function approximation tasks such as regression, classification, clustering, dimensionality reduction, etc. [13, 15, 16]. However, it is challenging to directly apply them to a decentralized multi-agent setting without any raw data sharing or aggregation. This is because the decision variables of kernel-based local objective functions are data-dependent and thus cannot be optimized in the absence of raw data exchange under the decentralized consensus framework. Moreover, standard kernel learning for big data incurs the curse of dimensionality issue when the number of training examples increases [15]. Furthermore, when computation cost is more affordable than the communication cost in the big data scenario, the communication cost of the iterative learning algorithms becomes the bottleneck for efficient distributed learning [7]. In this context, Chapter 2 advocates a new framework to

efficiently solve the nonlinear distributed learning problem under communication constraints and privacy concerns. The proposed framework is based on the random feature (RF) mapping, decentralized optimization through the alternating direction method of multipliers (ADMM), embedded with a communication-censoring technique. Theoretical analysis, as well as numerical tests for both synthetic and real datasets, confirm the effectiveness of the novel methods compared with existing methods [17, 18]. Note that to save communications for network multi-agent systems, we have also developed a class of event-triggered algorithms in [19, 20] and an infrequent communication scheme in [21], which can be adopted in the decentralized learning framework as well to reduce the communication load.

1.3.2 Quantized and communication-censored decentralized adaptive kernel learning

Chapter 3 focuses on online kernel learning over a decentralized network to perform function learning in an online fashion. The goal is to cope with online streaming data that exists in many real-life applications such as wearable devices that collect health statistics from sequential data or online spam detection [22, 23]. The RF mapping method is utilized to convert the non-parametric kernel learning problem into a fixed-length parametric one in the RF space. A novel online learning framework based on linearized ADMM is proposed to efficiently solve the online decentralized kernel learning problem. To further improve communication efficiency, two strategies (quantization and communication censoring) are incorporated into the decentralized online learning algorithm. Theoretical analysis shows that the proposed algorithm achieves the same level regret compared with the state-of-the-art, and the simulation results corroborate the learning effectiveness, communication, and computation efficiencies of the proposed methods [24].

1.3.3 Communication-efficient decentralized dynamic kernel learning

In addition to the online streaming data, in many cases, the dynamics underlying the streaming data are also varying. Thus the optimal function to be estimated also varies

over time with unknown dynamics, and no single estimate is universally good. Aiming to obtain the best actions "on the fly", Chapter 4 customizes the decentralized dynamic kernel learning problem as a decentralized dynamic optimization problem, with the utilization of RF mapping. Based on the reformulation, an ADMM-based algorithm with quantization and communication-censoring strategies is proposed to solve the dynamic kernel learning problem in a communication-efficient way. Further, the learning efficiency of the proposed algorithm is evaluated by both theoretical analysis and simulations on real datasets [25].

1.3.4 Deep kernel learning networks: A unified framework of learning nonlinear input-output maps

Neural networks and kernel methods are both capable of learning rich representations from nonlinear input-output maps. Despite their successes, there is a lack of understanding of their relationship and guidelines of which one should be applied for specific tasks. To fill such a gap, Chapter 5 takes the first step toward developing a unified framework of these two different learning techniques, which aims to not only unveil their underlying relationship but also enhance the explainability of neural networks. By leveraging the random feature mapping technique, kernel methods can be implemented by a two-layer neural network with specific activation functions, at a drastically reduced workload on weight training. The evident advantage in low complexity can be leveraged in a unified way to trade for enhanced learning performance, by expanding the network structure in depth and/or width. This motivates the development of a deep kernel learning (DKL) network by performing random feature mapping at each layer and training the last output layer only. To increase the representation power of DKL, we add multiple trainable paths to connect all hidden layers with the output layer. It leads to a deep kernel learning network structure with multiple learning paths (DKL-MLP), whose output functional is linear in the trainable multi-path model parameters. In this way, DKL-MLP benefits not only from the depth of DKL, but also from the (implicit) flexibility and computational advantage of RF-based multi-kernel learning. Theoretical analysis in terms of universality

Table 1.1: Main results and relevant publications (T is the total data size).

Chapter	Algorithms	Theoretical analysis	Papers	Status
2	DKLA, COKE	Linear convergence generalization error $\mathcal{O}(1/T)$	[17, 18]	published
3	ODKLA, QC-ODKLA	Static regret $\mathcal{O}(\sqrt{T})$	[24]	submitted
4	DDKL, QC-DDKL	Dynamic regret $\mathcal{O}(\sqrt{T})$	[25]	under preparation
5	DKL, DKL-MLP	Universal approximation	[26]	published
other	Event-triggered and energy-efficient algorithms	Asymptotic convergence	[19, 20] [21]	published

shows that the developed DKL and DKL-MLP algorithms can represent a wide variety of interesting functions with arbitrarily small errors and have no bad local minimum. In addition, simulations on both the classification and regression tasks corroborate that DKL-MLP enjoys both good generalization performance and low computational complexity [26].

1.4 Notational conventions

Throughout this dissertation, $\mathbb{R}$ denotes the set of real numbers. $\|\cdot\|_2$ denotes the Euclidean norm of vectors and $\|\cdot\|_F$ denotes the Frobenius norm of matrices. $|\cdot|$ denotes the cardinality of a set. For letters, $\mathbf{A}$ denotes a matrix, $\mathbf{a}$ denotes a vector, and a denotes a scalar. The identity matrix is denoted by $\boldsymbol{I}$ while $\mathbf{A}^\top$ stands for the transposition of matrix $\mathbf{A}$.

Chapter 2: Communication-Censored Decentralized Kernel Learning

2.1 Introduction

Kernel methods have proven to be successful in various learning tasks such as regression, classification, and dimensionality reduction, to name just a few [13,15]. This is because they utilize the so-called "kernel trick" to transform the originally nonlinear data into a higher dimensional implicit feature space so that the transformed data are linearly separable. Thus, many well-behaved linear learning algorithms are applicable in that high-dimensional space [13, 15, 16]. However, in the absence of any raw data sharing or aggregation, it is challenging to directly apply them to a decentralized multi-agent setting and solve them under the consensus optimization framework using algorithms such as decentralized alternating direction method of multipliers (ADMM) [27]. This is because decentralized learning relies on solving local optimization problems and then aggregating the updates on the local decision variables over the network through one-hop communications in an iterative manner [28]. Unfortunately, these decision variables of local objective functions resulted from the kernel trick are data-dependent and thus cannot be optimized in the absence of raw data exchange under the decentralized consensus framework.

There are several works applying kernel methods in decentralized learning for various applications under different settings [29–35]. These works, however, either assume that agents have access to their neighbors' observed raw data [29] or require agents to transmit their raw data to their neighbors [35] to ensure consensus through collaborative learning. These assumptions may not be valid in many practical applications that involve users' private data. Moreover, standard kernel learning for big data faces the curse of dimensionality when the number of training examples increases [15]. For example, in [30,32], the nonlinear

function learned at each node is represented as a weighted combination of kernel functions centered on its local observed data. As a result, each agent needs to transmit both the weights of kernel functions and its local data to its neighbors at every iterative step to guarantee consensus of the common prediction function. Thus, both the computation and communication resources are demanding in the distributed implementation. To alleviate the curse of dimensionality problem, [31] and [35] have developed compression techniques such as data selection and sparse subspace projection, respectively, but these techniques typically incur considerable extra computation, and still involve raw data exchange with no alleviation to the data privacy concern. Furthermore, when computation cost is more affordable than the communication in the big data scenario, communication cost of the iterative learning algorithms becomes the bottleneck for efficient distributed learning [7]. Therefore, it is crucial to design communication-efficient distributed kernel learning algorithms with data privacy protection.

2.1.1 Related work

This work lies at the intersection of non-parametric kernel methods, decentralized learning with batch-form data, and communication-efficient implementation. Related work to these three subjects is reviewed below.

Centralized kernel methods. Centralized kernel methods assume data are collected and processed by a single server and are known to suffer from the curse of dimensionality for large-scale learning tasks. To mitigate their computational complexity, various dimensionality reduction techniques are developed for both batch-form or online streaming learning, including stochastic approximation [36,37], restricting the number of function parameters [38–42], and approximating the kernel during training [43–52]. Among them, random feature (RF) mapping methods have gained popularity thanks to their ability to map the large-scale data into a RF space of much reduced dimension by approximating the kernel with a fixed (small) number of random features, which thus circumvents the curse of dimensionality problem [47,50–52]. Enforcing orthogonality on random features can greatly

reduce the error in kernel approximation [53,54], and the learning performance of RF-based methods is evaluated in [55–57].

Decentralized kernel learning. For the decentralized kernel learning problem relevant to our work [30–32, 35], gradient descent is conducted locally at each agent to update its learning model, followed by diffusion-based information exchange among agents. However, these methods either assume that agents have access to their neighbors' observed raw data or require agents to transmit their raw data to their neighbors to ensure convergence on the prediction function. For the problem studied in this article where the observed data are only locally available, these methods are not applicable since there are no common decision parameters for consensus without any raw data exchange. Moreover, these methods operate in the kernel space parameterized by training data, and still encounter the curse of dimensionality when the local dataset goes large. Though data selection [31] and subspace projection [35] are adopted to alleviate the curse of dimensionality problem, they typically require significant extra computational resources. RF mapping [50] offers a viable approach to overcome these issues, by having all agents map their datasets of various sizes onto the same RF space. For instance, [58] proposes a diffusion-based combine-then-adapt (CTA) method that achieves consensus on the model parameters in the RF space for the online learning problem, without the exchange of raw data. Though the batch-form counterpart of online CTA can be developed for off-line learning, the convergence speed of the diffusion-based method is relatively slow compared with higher-order methods such as ADMM [59].

Communication-efficient optimization. Communication-efficient algorithms for decentralized optimization and learning problems have attracted attention when data movement among computing nodes becomes a bottleneck due to the high latency and limited bandwidth of decentralized networks. To reduce the communication cost, one way is to transmit the compressed information by quantization [60–62] or sparsification [63–66]. However, these methods only reduce the required bandwidth at each communication round, not the number of rounds or the number of transmissions. Alternatively, some works randomly select a number of nodes for broadcasting/communication and operate asynchronous updating to reduce

the number of transmissions per iteration [7,67–72]. In contrast to random node selection, a more intuitive way is to evaluate the importance of a message in order to avoid unnecessary transmissions [10, 59, 73]. This is usually implemented by adopting a censoring scheme to adaptively decide if a message is informative enough to be transmitted during the iterative optimization process. Other efforts to improve the communication efficiency are made by accelerating the convergence speed of the iterative algorithm implementation [74–76].

2.1.2 Contributions

The present chapter develops communication-efficient decentralized kernel learning algorithms under the consensus optimization framework without any central coordination or raw data exchange among agents for built-in privacy protection. Relative to prior art, our contributions are summarized as follows.

- We first formulate the decentralized multi-agent kernel learning problem as a decentralized consensus optimization problem in the RF space. Since most machine learning scenarios can afford plenty computational capability but limited communication resources, we solve this problem with ADMM, which has shown fast convergence at the expense of relatively high computation cost per iteration [27]. To the best of our knowledge, this is the first work to solve the decentralized kernel learning in the RF space by ADMM without any raw data exchange. The key of our proposed Decentralized Kernel Learning via ADMM (DKLA) algorithm is to apply the RF mapping, which not only reduces the computational complexity but also enables consensus on a set of model parameters of fixed size in the RF space. In addition, since no raw data is exchanged among agents and the mapping from the original data space to the RF space is not one-to-one mapping, data privacy is protected to a certain point.

- To increase the communication efficiency, we further develop a COmmunication-censored KErnel learning (COKE) algorithm, which achieves desired learning performance given limited communication resources and energy supply. Specifically, we

devise a simple yet powerful censoring strategy to allow each user to autonomously skip unnecessary communications when its local update is not informative enough for transmission, without aid of a central coordinator. In this way, the communication efficiency can be boosted at almost no sacrifice of the learning performance. When the censoring strategy is absent, COKE degenerates to DKLA.

- In addition, we conduct theoretical analysis in terms of both functional convergence and generalization performance to provide guidelines for practical implementations of our proposed algorithms. We show that the individually learned functional at each agent through DKLA and COKE both converges to the optimal one at a linear rate under mild conditions. For the generalization performance, we show that $O(\sqrt{T}\log d_{\mathbf{K}}^{\lambda})$ features are sufficient to ensure $O(1/\sqrt{T})$ learning risk for the decentralized kernel ridge regression problem, where $d_{\mathbf{K}}^{\lambda}$ is the number of effective degrees of freedom that will be defined in Section 2.4.

- Finally, we test the performance of our proposed DKLA and COKE algorithms on both synthetic and real datasets. The results corroborate that both DKLA and COKE exhibit attractive learning performance and COKE is highly communication-efficient.

Organization. Section 2.2 formulates the problem of non-parametric learning and highlights the challenges in applying traditional kernel methods in the decentralized setting. Section 2.3 develops the decentralized kernel learning algorithms, including both DKLA and COKE. Section 2.4 presents the theoretical results and Section 2.5 reports the numerical tests using both synthetic data and real datasets. Concluding remarks are provided in Section 2.6.

2.2 Problem Statement

This section reviews basics of kernel-based learning and decentralized optimization, introduces notation, and provides technical background for our novel DKLA and COKE schemes.

Consider the N-agent network model described in Section 1.2, where each agent in the network only has access to its locally observed data composed of independently and identically distributed (i.i.d) input-label pairs $\{\mathbf{x}_{i,t}, y_{i,t}\}_{t=1}^{T_i}$ obeying an unknown probability distribution p on $\mathcal{X} \times \mathcal{Y}$, with $\mathbf{x}_{i,t} \in \mathbb{R}^d$ and $y_{i,t} \in \mathbb{R}$. The kernel learning task is to find a prediction function f that best describes the ensemble of all data from all agents. Suppose that f belongs to the reproducing kernel Hilbert space (RKHS) $\mathcal{H} := \{f | f(\mathbf{x}) = \sum_{t=1}^{\infty} \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t)\}$ induced by a positive semidefinite kernel $\kappa(\mathbf{x}, \mathbf{x}_t) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that measures the similarity between $\mathbf{x}$ and $\mathbf{x}_t$, for all $\mathbf{x}, \mathbf{x}_t \in \mathcal{X}$. In a decentralized setting with privacy concern, this means that each agent has to be able to learn the global function $f \in \mathcal{H}$ such that $y_{i,t} = f(\mathbf{x}_{i,t}) + e_{i,t}$ for $\{\{\mathbf{x}_{i,t}, y_{i,t}\}_{t=1}^{T_i}\}_{i=1}^{N}$, without exchange of any raw data and in the absence of a fusion center, where the error terms $e_{i,t}$ are minimized according to certain optimality metric.

To evaluate the learning performance, a nonnegative loss function $\ell(y, \hat{y})$ is utilized to measure the difference between the true label value y and the predicted value $\hat{y} = f(\mathbf{x})$. Some common loss functions include the quadratic loss $\ell(y, \hat{y}) = (y - \hat{y})^2$ for regression tasks, the hinge loss $\ell(y, \hat{y}) = \max(0, 1 - y\hat{y})$ and the logistic loss $\ell(y, \hat{y}) = \log(1 + e^{-y\hat{y}})$ for binary classification tasks. The above mentioned loss functions are all convex with respect to $\hat{y}$. The learning problem is then to minimize the expected risk of the prediction function:

$$R(f) = \int_{\mathcal{X} \times \mathcal{Y}} \ell(f(\mathbf{x}), y) dp(\mathbf{x}, y), \tag{2.1}$$

which indicates the generalization ability of f to new data.

However, the distribution p is unknown in most learning tasks. Therefore, minimizing $R(f)$ is not applicable. Instead, given the finite number of training examples, the problem

turns to minimizing the empirical risk:

$$\min_{f\in\mathcal{H}} \quad \hat{R}(f) := \sum_{i=1}^{N} \hat{R}_i(f), \tag{2.2}$$

where $\hat{R}_i(f)$ is the local empirical risk for agent i given by

$$\hat{R}_i(f) = \frac{1}{T_i}\sum_{t=1}^{T_i} \ell(f(\mathbf{x}_{i,t}), y_{i,t}) + \lambda_i \|f\|_{\mathcal{H}}^2, \tag{2.3}$$

with $\|\cdot\|_{\mathcal{H}}$ being the norm associated with $\mathcal{H}$, and $\lambda_i > 0$ being a regularization parameter that controls over-fitting.

The representer theorem states that the minimizer of a regularized empirical risk functional defined over a RKHS can be represented as a finite linear combination of kernel functions evaluated on the data pairs from the training dataset [77]. If $\{\{\mathbf{x}_{i,t}, y_{i,t}\}_{t=1}^{T_i}\}_{i=1}^{N}$ are centrally available at a fusion center, the minimizer of (2.2) admits

$$f^\star(\mathbf{x}) = \sum_{i=1}^{N}\sum_{t=1}^{T_i} \alpha_{i,t}\kappa(\mathbf{x}, \mathbf{x}_{i,t}) := \boldsymbol{\alpha}^\top \boldsymbol{\kappa}(\mathbf{x}), \tag{2.4}$$

where $\boldsymbol{\alpha} = [\alpha_{1,1}, \ldots, \alpha_{N,T_j}]^\top \in \mathbb{R}^T$ is the coefficient vector to be learned, $T = \sum_{i=1}^{N} T_i$ is the total number of samples, and $\boldsymbol{\kappa}(\mathbf{x}) = [\kappa(\mathbf{x}, \mathbf{x}_{1,1}), \ldots, \kappa(\mathbf{x}, \mathbf{x}_{N,T_N})]^\top \in \mathbb{R}^T$ is the kernel function parameterized by the global data $\mathbf{X}_T := \{\{\mathbf{x}_{i,t}\}_{t=1}^{T_i}\}_{i=1}^{N}$ from all agents, for any $\mathbf{x}$. In RKHS, since $\langle \kappa(\mathbf{x}_t, \mathbf{x}), \kappa(\mathbf{x}_\tau, \mathbf{x})\rangle_{\mathcal{H}} = \kappa(\mathbf{x}_t, \mathbf{x}_\tau)$, it yields $\|f\|_{\mathcal{H}}^2 = \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}$, where $\mathbf{K}$ is the $T \times T$ kernel matrix that measures the similarity between any two data points in $\mathbf{X}_T$. In this way, the local empirical risk (2.3) can be reformulated as a function of $\boldsymbol{\alpha}$:

$$\hat{R}_i(\boldsymbol{\alpha}) := \frac{1}{T_i}\sum_{t=1}^{T_i} \ell(f^\star(\mathbf{x}_{i,t}), y_{i,t}) + \lambda_i \|f^\star\|_{\mathcal{H}}^2 = \frac{1}{T_i}\sum_{t=1}^{T_i} \ell(\boldsymbol{\alpha}^\top \boldsymbol{\kappa}(\mathbf{x}_{i,t}), y_{i,t}) + \lambda_i \boldsymbol{\alpha}^\top \mathbf{K}\boldsymbol{\alpha}. \tag{2.5}$$

Accordingly, (2.2) becomes

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^T} \quad \sum_{i=1}^{N} \hat{R}_i(\boldsymbol{\alpha}). \tag{2.6}$$

Relating the decentralized kernel learning problem with the decentralized consensus optimization problem, solving (2.6) is equivalent to solving

$$\begin{aligned} \min_{\{\boldsymbol{\alpha}_i\in\mathbb{R}^T\}_{i=1}^N} \quad & \sum_{i=1}^{N} \hat{R}_i(\boldsymbol{\alpha}_i) \\ \text{s.t.} \quad & \boldsymbol{\alpha}_i = \boldsymbol{\alpha}_j, \qquad \forall i, \quad \forall j \in \mathcal{N}_i, \end{aligned} \tag{2.7}$$

where $\boldsymbol{\alpha}_i$ and $\boldsymbol{\alpha}_j$ are the local copies of the global decision variable $\boldsymbol{\alpha}$ at agent i and agent j, respectively. The problem can then be solved by ADMM [27] or other primal dual methods [78]. However, it is worth noting that (2.7) reveals a subtle yet profound difference from a general optimization problem for parametric learning. That is, each local function $\hat{R}_i$ depends on not only the global decision variable $\boldsymbol{\alpha}$, but also the global data $\mathbf{X}_T$ because of the kernel terms $\boldsymbol{\kappa}(\mathbf{x}_{i,t})$ and $\mathbf{K}$. As a result, solving the local objective for agent i requires raw data from all other agents to obtain $\boldsymbol{\kappa}(\mathbf{x}_{i,t})$ and $\mathbf{K}$, which contradicts the situation that private raw data are only locally available. Moreover, notice that $\boldsymbol{\alpha}_i$ is of the same size T as that of the ensemble dataset, which incurs the curse of dimensionality and insurmountable computational cost when T becomes large, even when the obstacle of making all the data available to all agents is not of concern.

To resolve this issue, an alternative formulation is to associate a local prediction model $\bar{f}_i \in \mathcal{H}$ with each agent i, with $\bar{f}_i^\star = \sum_{t=1}^{T_i} \bar{\alpha}_{i,t}\kappa(\mathbf{x}, \mathbf{x}_{i,t}) = \bar{\boldsymbol{\alpha}}_i^\top \boldsymbol{\kappa}_i(\mathbf{x})$ being the local optimal solution that only involves local data $\{\mathbf{x}_{i,t}\}_{t=1}^{T_i}$ [79]. Specifically, $\bar{\boldsymbol{\alpha}}_i = [\bar{\alpha}_{i,1}, \ldots, \bar{\alpha}_{i,T_i}] \in \mathbb{R}^{T_i}$, and $\boldsymbol{\kappa}_i(\mathbf{x}) = [\kappa(\mathbf{x}, \mathbf{x}_{i,1}), \ldots, \kappa(\mathbf{x}, \mathbf{x}_{i,T_i})]^\top \in \mathbb{R}^{T_i}$ is parameterized by the local data $\{\mathbf{x}_{i,t}\}_{t=1}^{T_i}$

only. In this way, the local cost function becomes

$$\hat{R}_i(\bar{\boldsymbol{\alpha}}_i) := \frac{1}{T_i}\sum_{t=1}^{T_i}\ell(\bar{f}_i^{\star}(\mathbf{x}_{i,t}), y_{i,t}) + \lambda_i\|\bar{f}_i^{\star}\|_{\mathcal{H}}^2 = \frac{1}{T_i}\sum_{t=1}^{T_i}\ell(\bar{\boldsymbol{\alpha}}_i^{\top}\boldsymbol{\kappa}_i(\mathbf{x}_{i,t}), y_{i,t}) + \lambda_i\bar{\boldsymbol{\alpha}}_i^{\top}\mathbf{K}_i\bar{\boldsymbol{\alpha}}_i, \quad (2.8)$$

where $\mathbf{K}_i$ is of size $T_i \times T_i$ and depends on local data only. With (2.8), the optimization problem (2.7) is then modified to

$$\begin{aligned} \min_{\{\bar{\boldsymbol{\alpha}}_i \in \mathbb{R}^{T_i}\}_{i=1}^N} \quad & \sum_{i=1}^{N}\hat{R}_i(\bar{\boldsymbol{\alpha}}_i) \\ \text{s.t.} \quad & \bar{f}_j(\mathbf{x}_{i,t}) = \bar{f}_i(\mathbf{x}_{i,t}), \qquad \forall i, \quad \forall j \in \mathcal{N}_i, \quad t = 1, \ldots, T_i, \end{aligned} \quad (2.9)$$

and can be solved distributedly by ADMM. Note that the consensus constraint is the learned prediction values $\bar{f}_i(\mathbf{x})$, not the parameters $\bar{\boldsymbol{\alpha}}_i$. This is because $\bar{\boldsymbol{\alpha}}_i$ are data-dependent and may have different sizes at different agents (the dimension of $\bar{\boldsymbol{\alpha}}_i$ equals to the number of training samples at agent i), and cannot be directly optimized through consensus.

Still, this method has four drawbacks. Firstly, it is necessary to associate a local learning model $\bar{f}_i$ to each agent i for the decentralized implementation. However, the local learning model $\bar{f}_i$ and the global optimal model f in (2.2) may not be the same because different local training data are used. Therefore, the optimization problem (2.9) is only an approximation of (2.2). Even with the equality constraint to minimize the gap between the decentralized learning output and the optimal centralized one, the approximation performance is not guaranteed. Besides, the functional consensus constraint still requires raw data exchange among agents in order for agent $j \in \mathcal{N}_i$ to be able to compute the values $\bar{f}_j(\mathbf{x}_{i,t})$ from agent i's data $\mathbf{x}_{i,t}$, for $i \neq j$. Apparently, this violates the privacy-protection requirement for practical applications. In addition, when T_i is large, both the storage and computational costs are high for each agent due to the curse of dimensionality problem at the local sites. Lastly, the frequent local communication is resource-consuming under communication constraints. To circumvent all these obstacles, the goal of this chapter is to

develop efficient decentralized algorithms that protect privacy and conserve communication resources.

2.3 Algorithm Development

In this section, we leverage the RF approximation and ADMM to develop our algorithms. We first introduce the RF mapping method. Then, we devise the DKLA algorithm that globally optimizes a shared learning model for the multi-agent system. Finally, we take into consideration of the limited communication resources in large-scale decentralized networks and develop the COKE algorithm. Both DKLA and COKE are computationally efficient and protect data privacy at the same time. Further, COKE is more communication efficient than DKLA and both are more communication efficient than CTA.

2.3.1 RF-based kernel learning

As stated in previous sections, standard kernel methods incur the curse of dimensionality issue when the data size grows large. To make kernel methods scalable for a large dataset, RF mapping is adopted for approximation by using the shift-invariance property of kernel functions [50].

For a shift-invariant kernel that satisfies $\kappa(\mathbf{x}_t, \mathbf{x}_\tau) = \kappa(\mathbf{x}_t - \mathbf{x}_\tau),\ \forall t,\ \forall \tau$, if $\kappa(\mathbf{x}_t - \mathbf{x}_\tau)$ is absolutely integrable, then its Fourier transform $p_\kappa(\boldsymbol{\omega})$ is guaranteed to be nonnegative ($p_\kappa(\boldsymbol{\omega}) \geq 0$), and hence can be viewed as its probability density function (pdf) when κ is scaled to satisfy $\kappa(0) = 1$ [80]. Therefore, we have

$$\kappa(\mathbf{x}_t, \mathbf{x}_\tau) = \int p_\kappa(\boldsymbol{\omega}) e^{j\boldsymbol{\omega}^\top(\mathbf{x}_t - \mathbf{x}_\tau)} d\boldsymbol{\omega} := \mathbb{E}_{\boldsymbol{\omega}}[e^{j\boldsymbol{\omega}^\top(\mathbf{x}_t - \mathbf{x}_\tau)}] = \mathbb{E}_{\boldsymbol{\omega}}[\phi(\mathbf{x}_t, \boldsymbol{\omega})\phi^*(\mathbf{x}_\tau, \boldsymbol{\omega})], \quad (2.10)$$

where $\mathbb{E}$ denotes the expectation operator, $\phi(\mathbf{x}, \boldsymbol{\omega}) := e^{j\boldsymbol{\omega}^\top \mathbf{x}}$ with $\boldsymbol{\omega} \in \mathbb{R}^d$, and $*$ is the complex conjugate operator. In (2.10), the first equality is the result of the Fourier inversion theorem, and the second equality arises by viewing $p_\kappa(\boldsymbol{\omega})$ as the pdf of $\boldsymbol{\omega}$. In this chapter,

we adopt a Gaussian kernel $\kappa(\mathbf{x}_t, \mathbf{x}_\tau) = \exp(-\|\mathbf{x}_\mathrm{t} - \mathbf{x}_\tau\|_2^2/(2\sigma^2))$, whose pdf is a normal distribution with $p_\kappa(\boldsymbol{\omega}) \sim \mathbf{N}(\mathbf{0}, \sigma^{-2}\mathbf{I})$.

The main idea of RF mapping is to randomly generate $\{\boldsymbol{\omega}_l\}_{l=1}^L$ from the distribution $p_\kappa(\boldsymbol{\omega})$ and approximate the kernel function $\kappa(\mathbf{x}_t, \mathbf{x}_\tau)$ by the sample average

$$\hat{\kappa}_L(\mathbf{x}_t, \mathbf{x}_\tau) := \frac{1}{L}\sum_{l=1}^{L} \phi(\mathbf{x}_t, \boldsymbol{\omega}_l)\phi^*(\mathbf{x}_\tau, \boldsymbol{\omega}_l) := \boldsymbol{\phi}_L^\dagger(\mathbf{x}_\tau)\boldsymbol{\phi}_L(\mathbf{x}_t), \tag{2.11}$$

where $\boldsymbol{\phi}_L(\mathbf{x}) := \sqrt{\frac{1}{L}}[\phi(\mathbf{x}, \boldsymbol{\omega}_1), \ldots, \phi(\mathbf{x}, \boldsymbol{\omega}_L)]^\top$ and $\dagger$ is the conjugate transpose operator.

The following real-valued mappings can be adopted to approximate $\kappa(\mathbf{x}_t, \mathbf{x}_\tau)$, both satisfying the condition $\mathbb{E}_{\boldsymbol{\omega}}[\phi_r(\mathbf{x}_t, \boldsymbol{\omega})^\top \phi_r(\mathbf{x}_\tau, \boldsymbol{\omega})] = \kappa(\mathbf{x}_t, \mathbf{x}_\tau)$ [50]:

$$\phi_r(\mathbf{x}, \boldsymbol{\omega}) = [\cos(\boldsymbol{\omega}^\top \mathbf{x}), \sin(\boldsymbol{\omega}^\top \mathbf{x})]^\top, \tag{2.12}$$

$$\phi_r(\mathbf{x}, \boldsymbol{\omega}) = \sqrt{2}\cos(\boldsymbol{\omega}^\top \mathbf{x} + b), \tag{2.13}$$

where b is drawn uniformly from $[0, 2\pi]$.

With the real-valued RF mapping, the minimizer of (2.2) then admits the following form:

$$\hat{f}^\star(\mathbf{x}) = \sum_{i=1}^{N}\sum_{t=1}^{T_i} \alpha_{i,t}\boldsymbol{\phi}_L^\top(\mathbf{x}_{i,t})\boldsymbol{\phi}_L(\mathbf{x}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}_L(\mathbf{x}), \tag{2.14}$$

where $\boldsymbol{\theta}^\top := \sum_{i=1}^{N}\sum_{t=1}^{T_i} \alpha_{i,t}\boldsymbol{\phi}_L^\top(\mathbf{x}_{i,t})$ denotes the new decision vector to be learned in the RF space and $\boldsymbol{\phi}_L(\mathbf{x}) = \sqrt{\frac{1}{L}}[\phi_r(\mathbf{x}, \boldsymbol{\omega}_1), \ldots, \phi_r(\mathbf{x}, \boldsymbol{\omega}_L)]^\top$. If (2.12) is adopted, then $\boldsymbol{\phi}_L(\mathbf{x})$ and $\boldsymbol{\theta}$ are of size $2L$. Otherwise, if (2.13) is adopted, then $\boldsymbol{\phi}_L(\mathbf{x})$ and $\boldsymbol{\theta}$ are of size L. In either case, the size of $\boldsymbol{\theta}$ is fixed and does not increase with the number of data samples.

2.3.2 DKLA: Decentralized kernel learning via ADMM

Consider the decentralized kernel learning problem described in Section 2.2 and adopt the RF mapping described in Section 2.3.1. Let all agents in the network have the same set of random features, i.e., $\{\boldsymbol{\omega}_l\}_{l=1}^L$. Plugging (2.14) into the local cost function $\hat{R}_i(f)$ in (2.3) gives

$$\hat{R}_i(\boldsymbol{\theta}) := \frac{1}{T_i}\sum_{t=1}^{T_i}\ell(\hat{f}^\star(\mathbf{x}_{i,t}),y_{i,t})+\lambda_i\|\hat{f}^\star\|_{\mathcal{H}}^2=\frac{1}{T_i}\sum_{t=1}^{T_i}\ell(\boldsymbol{\theta}^\top\boldsymbol{\phi}_L(\mathbf{x}_{i,t}),y_{i,t})+\lambda_i\|\boldsymbol{\theta}\|_2^2. \quad (2.15)$$

In (2.15), we have

$$\begin{aligned}\|\boldsymbol{\theta}\|_2^2 :&= (\sum_{i=1}^N\sum_{t=1}^{T_i}\alpha_{i,t}\boldsymbol{\phi}_L^\top(\mathbf{x}_{i,t}))(\sum_{j=1}^N\sum_{\tau=1}^{T_i}\alpha_{j,\tau}\boldsymbol{\phi}_L(\mathbf{x}_{j,\tau}))\\ &=\sum_{i=1}^N\sum_{t=1}^{T_i}\sum_{j=1}^N\sum_{\tau=1}^{T_i}\alpha_{i,t}\alpha_{j,\tau}\kappa(\mathbf{x}_{i,t},\mathbf{x}_{j,\tau}):=\|\hat{f}^\star\|_{\mathcal{H}}^2.\end{aligned}$$

Therefore, with RF mapping, the centralized benchmark (2.2) becomes

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^L}\quad\sum_{i=1}^N\hat{R}_i(\boldsymbol{\theta}). \quad (2.16)$$

Here for notation simplicity, we denote the size of $\boldsymbol{\theta}$ by $L\times 1$, which can be achieved by adopting the real-valued mapping in (2.13). Adopting an alternative mapping such as (2.12) only changes the size of $\boldsymbol{\theta}$ while the algorithm development is the same. RF mapping is essential because it results in a common optimization parameter $\boldsymbol{\theta}$ of fixed size for all agents.

To solve (2.16) in a decentralized manner, we associate a model parameter $\boldsymbol{\theta}_i$ with each agent i and enforce the consensus constraint on neighboring agents i and j using an auxiliary variable $\boldsymbol{z}_{ij}$. Specifically, the RF-based decentralized kernel learning problem is formulated

to jointly minimize the following objective function:

$$\begin{aligned} \min_{\{\boldsymbol{\theta}_i\in\mathbb{R}^L\},\{\boldsymbol{z}_{ij}\in\mathbb{R}^L\}} \quad & \sum_{i=1}^{N} \hat{R}_i(\boldsymbol{\theta}_i) \\ \text{s.t.} \quad & \boldsymbol{\theta}_i = \boldsymbol{z}_{ij},\ \boldsymbol{\theta}_j = \boldsymbol{z}_{ij}, \qquad \forall i, \forall j \in \mathcal{N}_i. \end{aligned} \tag{2.17}$$

Note that the new decision variables $\boldsymbol{\theta}_i$ to be optimized are local copies of the global optimization parameter $\boldsymbol{\theta}$ and are of the same size for all agents. On the contrary, the decision variables $\bar{\boldsymbol{\alpha}}_i$ in (2.9) are data-dependent and may have different sizes. In addition, the size of $\boldsymbol{\theta}$ is L, which can be much smaller than that of $\boldsymbol{\alpha}$ (whose size equals to T) in (2.6). For big data scenarios where $L \ll T$, RF mapping greatly reduces the computational complexity. Moreover, as shown in the following, the updating of $\boldsymbol{\theta}$ does not involve any raw data exchange and the RF mapping from $\mathbf{x}$ to $\boldsymbol{\phi}_L(\mathbf{x})$ is not one-to-one mapping, therefore provides raw data privacy protection. Further, it is easy to set the regularization parameters λ_i to control over-fitting. Specifically, since the parameters $\boldsymbol{\theta}_i$ are of the same length among agents, we can set them to be $\lambda_i = \frac{1}{N}\lambda, \forall i$, where λ is the corresponding over-fitting control parameter assuming all data are collected at a center. In contrast, the regularization parameters λ_i in (2.5) depend on local data and need to satisfy $\lambda = \sum_{i=1}^{N} \lambda_i$, which is relatively difficult to tune in a large-scale network.

In the constraint, $\boldsymbol{\theta}_i$ are separable when $\boldsymbol{z}_{ij}$ are fixed, and vice versa. Therefore, (2.17) can be solved by ADMM. Following [27], we develop the DKLA algorithm where each agent updates its local primal variable $\boldsymbol{\theta}_i$ and local dual variable $\boldsymbol{\gamma}_i$ by

$$\boldsymbol{\theta}_i^k := \arg\min_{\boldsymbol{\theta}_i} \ \Big\{ \hat{R}_i(\boldsymbol{\theta}_i) + \rho|\mathcal{N}_i|\|\boldsymbol{\theta}_i\|_2^2 + \boldsymbol{\theta}_i^\top \Big[\boldsymbol{\gamma}_i^{k-1} - \rho \sum_{j\in\mathcal{N}_i} (\boldsymbol{\theta}_i^{k-1} + \boldsymbol{\theta}_j^{k-1}\ ) \Big] \Big\}, \tag{2.18a}$$

$$\boldsymbol{\gamma}_i^k = \boldsymbol{\gamma}_i^{k-1} + \rho \sum_{j\in\mathcal{N}_i} \Big(\boldsymbol{\theta}_i^k - \boldsymbol{\theta}_j^k \Big), \tag{2.18b}$$

where $|\mathcal{N}_i|$ is the cardinality of $\mathcal{N}_i$ and ρ is the stepsize. The auxiliary variable $\boldsymbol{z}_{ij}$ can be

Algorithm 1 DKLA Run at Agent i

Require: Kernel κ, the number of random features L, and λ to control over-fitting; initialize local variables to $\boldsymbol{\theta}_i^0 = \mathbf{0}$, $\boldsymbol{\gamma}_i^0 = \mathbf{0}$; set step size $\rho > 0$.

1: Draw L i.i.d. samples $\{\boldsymbol{\omega}_l\}_{l=1}^L$ from $p_\kappa(\boldsymbol{\omega})$ according to a common random seed.
2: Construct $\{\boldsymbol{\phi}_L(\mathbf{x}_{i,t})\}_{t=1}^{T_i}$ using the random features $\{\boldsymbol{\omega}_l\}_{l=1}^L$ via (2.12) or (2.13).
3: **for** iterations $k = 1, 2, \cdots$ **do**
4: Update local variable $\boldsymbol{\theta}_i^k$ by (2.18a);
5: Transmit $\boldsymbol{\theta}_i^k$ to all neighbor j ($j \in \mathcal{N}_i$) and receive $\boldsymbol{\theta}_j^k$ from all neighbor j;
6: Update local dual variable $\boldsymbol{\gamma}_i^k$ by (2.18b).
7: **end for**

written as a function of $\boldsymbol{\theta}_i$ and then canceled out. Interested readers are referred to [27] for detailed derivation. The learning algorithm DKLA is outlined in Algorithm 1. Note that the random features need to be common to all agents, hence, in step 1, we restrict them to be drawn according to a common random seed. Algorithm 1 is fully decentralized since the updates of $\boldsymbol{\theta}_i$ and $\boldsymbol{\gamma}_i$ depend only on local and neighboring information.

2.3.3 COKE: Communication-censored decentralized kernel learning

From Sections 2.3.1 and 2.3.2, we can see that decentralized kernel learning in the RF space under the consensus optimization framework has much reduced computational complexity, thanks to the RF mapping technique that transforms the learning model into a smaller RF space. In this subsection, we consider the case when the communication resource is limited and we aim to further reduce the communication cost of DKLA. To start, we notice that in Algorithm 1, each agent i ($i \in \mathcal{N}$) maintains $2 + |\mathcal{N}_i|$ local variables at iteration k, i.e., its local primal variable $\boldsymbol{\theta}_i^k$, local dual variable $\boldsymbol{\gamma}_i^k$ and $|\mathcal{N}_i|$ state variables $\boldsymbol{\theta}_j^k$ received from its neighbors. While the dual variable $\boldsymbol{\gamma}_i^k$ is kept locally for agent i, the transmission of its updated local variable $\boldsymbol{\theta}_i^k$ to its one-hop neighbors happens in every iteration, which consumes a large amount of communication bandwidth and energy along iterations for large-scale networks. In order to improve the communication efficiency, we develop the COKE algorithm by employing a censoring function at each agent to decide if a local update is informative enough to be transmitted.

To evaluate the importance of a local update at iteration k for agent i ($i \in \mathcal{N}$), we introduce a new state variable $\hat{\boldsymbol{\theta}}_i^{k-1}$ to record agent i's latest broadcast primal variable up to time $k-1$. Then, at iteration k, we define the difference between agent i's current state $\boldsymbol{\theta}_i^k$ and its previously transmitted state $\hat{\boldsymbol{\theta}}_i^{k-1}$ as

$$\boldsymbol{\xi}_i^k = \hat{\boldsymbol{\theta}}_i^{k-1} - \boldsymbol{\theta}_i^k, \tag{2.19}$$

and choose a censoring function as

$$H_i(k, \boldsymbol{\xi}_i^k) = \|\boldsymbol{\xi}_i^k\|_2 - h_i(k), \tag{2.20}$$

where $\{h_i(k)\}$ is a non-increasing non-negative sequence. A typical choice for the censoring function is $H_i(k, \boldsymbol{\xi}_i^k) = \|\boldsymbol{\xi}_i^k\|_2 - v\mu^k$, where $\mu \in (0, 1)$ and $v > 0$ are constants. When $H_i(k, \boldsymbol{\xi}_i^k) < 0$, $\boldsymbol{\theta}_i^k$ is deemed not informative enough, and hence will not be transmitted to its neighbors.

When executing the COKE algorithm, each agent i maintains $3 + |\mathcal{N}_i|$ local variables at each iteration k. Comparing with the DKLA update in (2.18), the additional local variable is the state variable $\hat{\boldsymbol{\theta}}_i^k$ that records its latest broadcast primal variable up to time k. Moreover, the $|\mathcal{N}_i|$ state variables from its neighbors are $\hat{\boldsymbol{\theta}}_j^k$ that record the latest received primal variables from its neighbors, instead of the timely updated and broadcast variables $\boldsymbol{\theta}_j^k$ of its neighbors $j \in \mathcal{N}_i$. While in COKE, each agent computes local updates at every step, its transmission to neighbors does not always occur, but is determined by the censoring criterion (2.20). To be specific, at each iteration k, if $H_i(k, \boldsymbol{\xi}_i^k) \geq 0$, then $\hat{\boldsymbol{\theta}}_i^k = \boldsymbol{\theta}_i^k$, and agent i is allowed to transmit its local primal variable $\boldsymbol{\theta}_i^k$ to its neighbors. Otherwise, $\hat{\boldsymbol{\theta}}_i^k = \hat{\boldsymbol{\theta}}_i^{k-1}$ and no information is transmitted. If agent i receives $\boldsymbol{\theta}_j^k$ from any neighbor j, then that neighbor's state variable kept by agent i becomes $\hat{\boldsymbol{\theta}}_j^k = \boldsymbol{\theta}_j^k$, otherwise, $\hat{\boldsymbol{\theta}}_j^k = \hat{\boldsymbol{\theta}}_j^{k-1}$.

Algorithm 2 COKE Run at Agent i

Require: Kernel κ, the number of random features L, the censoring thresholds $\{h_i(k)\}$, and λ to control over-fitting; initialize local variables to $\boldsymbol{\theta}_i^0 = \mathbf{0}$, $\hat{\boldsymbol{\theta}}_i^0 = \mathbf{0}$, $\hat{\boldsymbol{\theta}}_j^0 = \mathbf{0}$ for $j \in \mathcal{N}_i$, $\boldsymbol{\gamma}_i^0 = \mathbf{0}$; set step size $\rho > 0$.

Draw L i.i.d. samples $\{\boldsymbol{\omega}_l\}_{l=1}^L$ from $p_\kappa(\boldsymbol{\omega})$ according to a common random seed.
Construct $\{\boldsymbol{\phi}_L(\mathbf{x}_{i,t})\}_{t=1}^{T_i}$ using the random features $\{\boldsymbol{\omega}_l\}_{l=1}^L$ via (2.12) or (2.13).
for iterations $k = 1, 2, \cdots$ **do**
  Update local variable $\boldsymbol{\theta}_i^k$ by (2.21a);
  Compute $\boldsymbol{\xi}_i^k = \hat{\boldsymbol{\theta}}_i^{k-1} - \boldsymbol{\theta}_i^k$;
  If $H_i(k, \boldsymbol{\xi}_i^k) = \|\boldsymbol{\xi}_i^k\|_2 - h_i(k) \geq 0$, transmit $\boldsymbol{\theta}_i^k$ to neighbors and let $\hat{\boldsymbol{\theta}}_i^k = \boldsymbol{\theta}_i^k$; else do not transmit and let $\hat{\boldsymbol{\theta}}_i^k = \hat{\boldsymbol{\theta}}_i^{k-1}$;
  If receives $\boldsymbol{\theta}_j^k$ from neighbor j, let $\hat{\boldsymbol{\theta}}_j^k = \boldsymbol{\theta}_j^k$; else let $\hat{\boldsymbol{\theta}}_j^k = \hat{\boldsymbol{\theta}}_j^{k-1}$;
  Update local dual variable $\boldsymbol{\gamma}_i^k$ by (2.21b).
end for

Consequently, agent i's local parameters are updated as follows:

$$\boldsymbol{\theta}_i^k := \arg\min_{\boldsymbol{\theta}_i} \left\{ \hat{R}_i(\boldsymbol{\theta}_i) + \rho|\mathcal{N}_i|\|\boldsymbol{\theta}_i\|_2^2 + \boldsymbol{\theta}_i^\top \Big[\boldsymbol{\gamma}_i^{k-1} - \rho \sum_{j \in \mathcal{N}_i} \big(\hat{\boldsymbol{\theta}}_i^{k-1} + \hat{\boldsymbol{\theta}}_j^{k-1} \big) \Big] \right\}, \tag{2.21a}$$

$$\boldsymbol{\gamma}_i^k = \boldsymbol{\gamma}_i^{k-1} + \rho \sum_{j \in \mathcal{N}_i} \Big(\hat{\boldsymbol{\theta}}_i^k - \hat{\boldsymbol{\theta}}_j^k \Big), \tag{2.21b}$$

with a censoring step conducted between (2.21a) and (2.21b). We outline the COKE algorithm in Algorithm 2.

The key feature of COKE is that agent i's local variables $\boldsymbol{\theta}_i^k$ and $\boldsymbol{\gamma}_i^k$ are updated all the time, but the transmission of $\boldsymbol{\theta}_i^k$ occurs only when the censoring condition is met. By skipping unnecessary transmissions, the communication efficiency of COKE is improved. It is obvious that large $\{h_i(k)\}$ saves more communication but may lead to divergence from the optimal solution $\boldsymbol{\theta}^\star$ of (2.16), while small $\{h_i(k)\}$ does not contribute much to communication saving. Noticeably, DKLA is a special case of COKE when the communication censoring strategy is absent by setting $h_i(k) = 0, \forall i, k$.

2.4 Theoretical Guarantees

In this section, we perform theoretical analyses to address two questions related to the convergence properties of DKLA and COKE algorithms. First, do they converge to the globally optimal point, and if so, at what rate? Second, what is their achieved generalization performance in learning? Since DKLA is a special case of COKE, the analytic results of COKE, especially the second one, extend to DKLA straightforwardly. For theoretical analysis, we make the following assumptions.

Assumption 1. *The local cost functions $\hat{R}_i$ are strongly convex with constants $m_{\hat{R}_i} > 0$ such that $\forall i \in \mathcal{N}$, $\langle \nabla \hat{R}_i(\tilde{\boldsymbol{\theta}}_a) - \nabla \hat{R}_i(\tilde{\boldsymbol{\theta}}_b), \tilde{\boldsymbol{\theta}}_a - \tilde{\boldsymbol{\theta}}_b \rangle \geq m_{\hat{R}_i} \|\tilde{\boldsymbol{\theta}}_a - \tilde{\boldsymbol{\theta}}_b\|_2^2$, for any $\tilde{\boldsymbol{\theta}}_a, \tilde{\boldsymbol{\theta}}_b \in \mathbb{R}^L$. The minimum convexity constant is $m_{\hat{R}} := \min_i m_{\hat{R}_i}$. The gradients of the local cost functions are Lipschitz continuous with constants $M_{\hat{R}_i} > 0, \forall\ i$. That is, $\|\nabla \hat{R}_i(\tilde{\boldsymbol{\theta}}_a) - \nabla \hat{R}_i(\tilde{\boldsymbol{\theta}}_b)\|_2 \leq M_{\hat{R}_i} \|\tilde{\boldsymbol{\theta}}_a - \tilde{\boldsymbol{\theta}}_b\|_2$ for any agent i given any $\tilde{\boldsymbol{\theta}}_a, \tilde{\boldsymbol{\theta}}_b \in \mathbb{R}^L$. The maximum Lipschitz constant is $M_{\hat{R}} := \max_i M_{\hat{R}_i}$.*

Assumption 2. *The number of training samples of different agents is of the same order of magnitude, i.e., $\frac{\max_i T_i - \min_i T_i}{\min_i T_i} < 10, \forall i \in \mathcal{N}$.*

Assumption 3. *There exists $f_{\mathcal{H}} \in \mathcal{H}$, such that for all estimators $f \in \mathcal{H}$, $\mathcal{E}(f_{\mathcal{H}}) \leq \mathcal{E}(f)$, where $\mathcal{E}(f) := \mathbb{E}_p \left[\ell(f(\mathbf{x}), y)\right]$ is the expected risk to measure the generalization ability of the estimator f.*

Assumption 1 is standard for decentralized optimization over decentralized networks [27], Assumption 3 is standard in generalization performance analysis of kernel learning [57], and Assumption 2 is enforced to exclude the case of extremely unbalanced data distributed over the network.

2.4.1 Linear convergence of DKLA and COKE

We first establish that DKLA enables agents in the decentralized network to reach consensus on the prediction function at a linear rate. We then show that when the censoring function is properly chosen and the penalty parameter satisfies certain conditions, COKE also guarantees that the individually learned functional on the same sample linearly converges to the optimal solution.

Theorem 1. ***[Linear convergence of DKLA]*** *Initialize the dual variables as* $\boldsymbol{\gamma}_i^0 = \mathbf{0}$, $\forall i$, *with Assumption 1 and Assumption 2, the learned functional at each agent through DKLA is R-linearly convergent to the optimal functional* $\hat{f}_{\boldsymbol{\theta}^\star}(\mathbf{x}) := (\boldsymbol{\theta}^\star)^\top \boldsymbol{\phi}_L(\mathbf{x})$ *for any* $\mathbf{x} \in \mathcal{X}$, *where* $\boldsymbol{\theta}^\star$ *denotes the optimal solution to* (2.16) *obtained in the centralized case. That is,*

$$\lim_{k\to\infty} \hat{f}_{\boldsymbol{\theta}_i^k}(\mathbf{x}) = \hat{f}_{\boldsymbol{\theta}^\star}(\mathbf{x}), \forall i. \tag{2.22}$$

Proof. See Appendix 2.7.1. ■

Theorem 2. ***[Linear convergence of COKE]*** *Initialize the dual variables as* $\boldsymbol{\gamma}_i^0 = \mathbf{0}$, $\forall i$, *set the censoring thresholds to be* $h(k) = v\mu^k$, *with* $v > 0$ *and* $\mu \in (0, 1)$, *and choose the penalty parameter* ρ *such that*

$$0 < \rho < \min\left\{ \frac{4m_{\hat{R}}}{\eta_1}, \frac{(\nu-1)\tilde{\sigma}^2_{\min}(\mathbf{S}_-)}{\nu\eta_3\tilde{\sigma}^2_{\max}(\mathbf{S}_+)}, \left(\frac{\eta_1}{4} + \frac{\eta_2\tilde{\sigma}^2_{\max}(\mathbf{S}_+)}{8}\right)^{-1}\left(m_{\hat{R}} - \frac{\eta_3\nu M^2_{\hat{R}}}{\tilde{\sigma}^2_{\min}(\mathbf{S}_-)}\right)\right\}, \tag{2.23}$$

where $\eta_1 > 0$, $\eta_2 > 0$, $\eta_3 > 0$ *and* $\nu > 1$ *are arbitrary constants,* $m_{\hat{R}}$ *and* $M_{\hat{R}}$ *are the minimum strong convexity constant of the local cost functions and the maximum Lipschitz constant of the local gradients, respectively.* $\tilde{\sigma}_{\max}(\mathbf{S}_+)$ *and* $\tilde{\sigma}_{\min}(\mathbf{S}_-)$ *are the maximum singular value of the unsigned incidence matrix* $\mathbf{S}_+$ *and the minimum non-zero singular value of the signed incidence matrix* $\mathbf{S}_-$ *of the network, respectively. Then, with Assumption 1 and Assumption 2, the learned functional at each agent through COKE is R-linearly convergent*

to the optimal one $\hat{f}_{\boldsymbol{\theta}^\star}(\mathbf{x}) := (\boldsymbol{\theta}^\star)^\top \boldsymbol{\phi}_L(\mathbf{x})$ *for any* $\mathbf{x} \in \mathcal{X}$*, where* $\boldsymbol{\theta}^\star$ *denotes the optimal solution to* (2.16) *obtained in the centralized case. That is,*

$$\lim_{k\to\infty} \hat{f}_{\boldsymbol{\theta}_i^k}(\mathbf{x}) = \hat{f}_{\boldsymbol{\theta}^\star}(\mathbf{x}), \forall i. \tag{2.24}$$

Proof. See Appendix 2.7.1. ■

Remark 1. It should be noted that the kernel transformation with RF mapping is essential in enabling convex consensus formulation with convergence guarantee. For example, in a regular optimization problem with a local cost function $(y - f(\mathbf{x}))^2$, even if it is quadratic, the nonlinear function $f(\mathbf{x})$ inside destroys the convexity. In contrast, with RF mapping, $f(\mathbf{x})$ of any form is expressed as a linear function of $\boldsymbol{\theta}$, and hence the local cost function is guaranteed to be convex. For decentralized kernel learning, many widely-adopted loss functions result in (strongly) convex local objective functions in the RF space, such as the quadratic loss in a regression problem and logistic loss in a classification problem.

Remark 2. For Theorem 2, notice that choosing larger v and μ in the design of the censoring thresholds in COKE leads to less communication per iteration at the expense of possible performance degradation, whereas smaller v and μ may not contribute much to communication saving. However, it is challenging to acquire an explicit tradeoff between communication cost and steady-state accuracy, since the designed censoring thresholds do not have an explicit relationship with the update of the model parameter.

The above theorems establish the exact convergence of the functional learned in the multi-agent system for the decentralized kernel regression problem via DKLA and COKE. Different from previous works [34, 35], our analytic results are obtained by converting the non-parametric data-dependent learning model into a parametric data-independent model in the RF space and solved under the consensus optimization framework. In this way, we not only reduce the computational complexity of the standard kernel methods and make the RF-based kernel methods scalable to large-size datasets, but also protect data privacy since no raw data exchange among agents is required and the RF mapping is not one-to-one

mapping. RF mapping is crucial in our algorithms, with which we are able to show the linear convergence of the functional by showing the linear convergence of the iteratively updated decision variables in the RF space; see Appendix A for more details.

2.4.2 Generalization property of COKE

The ultimate goal of decentralized learning is to find a function that generalizes well for the ensemble of all data from all agents. To evaluate the generalization property of the predictive function learned by COKE, we are then interested in bounding the difference between the expected risk of the predictive function learned by COKE at the k-th iteration, defined as $\mathcal{E}(\hat{f}^k) := \sum_{i=1}^N \mathcal{E}_i(\hat{f}^k_{\boldsymbol{\theta}_i^k}) := \sum_{i=1}^N \mathbb{E}_p[(y-\boldsymbol{\theta}_i^{k\top}\boldsymbol{\phi}_L(\mathbf{x}))^2]$, and the expected risk $\mathcal{E}(f_{\mathcal{H}})$ in the RKHS. This is different from bounding the approximation error between the kernel κ and the approximated $\hat{\kappa}_L$ by L random features as in the literature [50,81,82]. As DKLA is a special case of COKE, the generalization performance of COKE can be extended to DKLA straightforwardly.

To illustrate our finding, we focus on the kernel regression problem whose loss function is least squares, i.e., $\ell(y, f(\mathbf{x})) = (y - f(\mathbf{x}))^2$. With RF mapping, the objective function (2.16) of the regression problem can be formulated as

$$\hat{R}(\boldsymbol{\theta}) = \sum_{i=1}^N \hat{R}_i(\boldsymbol{\theta}) = \sum_{i=1}^N \left(\frac{1}{T_i} \|\mathbf{y}_i - \boldsymbol{\Phi}_L^{i\top}\boldsymbol{\theta}\|_2^2 + \frac{\lambda}{N}\|\boldsymbol{\theta}\|_2^2 \right), \tag{2.25}$$

where $\mathbf{y}_i = [y_{i,1}, \ldots, y_{i,T_i}]^\top \in \mathbb{R}^{T_i \times 1}$, $\boldsymbol{\Phi}_L^i = [\boldsymbol{\phi}_L(\mathbf{x}_{i,1}), \ldots, \boldsymbol{\phi}_L(\mathbf{x}_{i,T_i})] \in \mathbb{R}^{L \times T_i}$, and $\boldsymbol{\phi}_L(\mathbf{x}_{i,t})$ is the data mapped to the RF space.

The optimal solution of (2.25) is given in closed form by

$$\boldsymbol{\theta}^\star = (\tilde{\boldsymbol{\Phi}}^\top \tilde{\boldsymbol{\Phi}} + \lambda \mathbf{I})^{-1} \tilde{\boldsymbol{\Phi}}^\top \tilde{\mathbf{y}}, \tag{2.26}$$

where $\tilde{\boldsymbol{\Phi}} = [\tilde{\boldsymbol{\Phi}}_L^1, \ldots, \tilde{\boldsymbol{\Phi}}_L^N]^\top \in \mathbb{R}^{T \times L}$ with $\tilde{\boldsymbol{\Phi}}_L^i = \frac{1}{\sqrt{T_i}} \boldsymbol{\Phi}_L^i, \forall i \in \mathcal{N}$, and $\tilde{\mathbf{y}} = [\tilde{\mathbf{y}}_1; \ldots; \tilde{\mathbf{y}}_N] \in$

$\mathbb{R}^{T\times 1}$ with $\tilde{\mathbf{y}}_i = \frac{1}{\sqrt{T_i}}\mathbf{y}_i, \forall i \in \mathcal{N}$. The optimal prediction model is then expressed by

$$\hat{f}_{\boldsymbol{\theta}^\star}(\mathbf{x}) = \boldsymbol{\theta}^{\star\top}\boldsymbol{\phi}_L(\mathbf{x}). \tag{2.27}$$

In the following theorem, we give a general result of the generalization performance of the predictive function learned by COKE for the kernel regression problem, which is built on the linear convergence result given in Theorem 2 and taking into account of the number of random features adopted.

Theorem 3. *Let $\lambda_{\mathbf{K}}$ be the largest eigenvalue of the kernel matrix $\mathbf{K}$ constructed by all data, $\mathbf{X}_T$, and choose the regularization parameter $\lambda < \lambda_{\mathbf{K}}/T$ so as to control overfitting. Under the Assumptions 1 - 3, with the censoring function and other parameters given in Theorem 2, for all $\delta_p \in (0,1)$ and $\|f\|_{\mathcal{H}} \leq 1$, if the number of random features L satisfies*

$$L \geq \frac{1}{\lambda}(\frac{1}{\epsilon^2} + \frac{2}{3\epsilon})\log\frac{16 d_{\mathbf{K}}^{\lambda}}{\delta_p},$$

then with probability at least $1-\delta_p$, the excess risk of $\mathcal{E}(\hat{f}^k)$ obtained by Algorithm 2 converges to an upper bound, i.e.,

$$\lim_{k\to\infty}(\mathcal{E}(\hat{f}^k) - \mathcal{E}(f_{\mathcal{H}})) \leq 3\lambda + O(\frac{1}{\sqrt{T}}), \tag{2.28}$$

where $\epsilon \in (0,1)$, and $d_{\mathbf{K}}^{\lambda} := \mathrm{Tr}(\mathbf{K}(\mathbf{K}+\lambda T\mathbf{I})^{-1})$ is the number of effective degrees of freedom that is known to be an indicator of the number of independent parameters in a learning problem [83].

Proof. See Appendix 2.7.2. ■

Theorem 3 states the tradeoff between the computational efficiency and the statistical efficiency through the regularization parameter λ, effective dimension $d_{\mathbf{K}}^{\lambda}$, and the number of random features adopted. We can see that to bound the excess risk with a higher probability,

we need more random features, which results in a higher computational complexity. The regularization parameter is usually determined by the number of training data and one common practice is to set $\lambda = O(1/\sqrt{T})$ for the regression problem [84]. Therefore, with $O(\sqrt{T}\log d_{\mathbf{K}}^{\lambda})$ features, COKE achieves a learning risk of $O(1/\sqrt{T})$ at a linear rate. We also notice that different sampling strategies affect the number of random features required to achieve a given generalization error. For example, importance sampling is studied for the centralized kernel learning in RF space in [57]. Interested readers are referred to [57] and references therein.

2.5 Experiments

This section evaluates the performance of our COKE algorithm in regression tasks using both synthetic and real-world datasets. Since we consider the case that data are only locally available and cannot be shared among agents, the following RF-based methods are used to benchmark our COKE algorithm.

CTA. This is a form of diffusion-based technique where all agents first construct their RF-mapped data $\{\boldsymbol{\phi}_L(\mathbf{x}_{i,t})\}_{t=1}^{T_i}$, for $t = 1, \ldots, T_i, \forall i$, using the same random features $\{\boldsymbol{\omega}_l\}_{l=1}^{L}$ as DKLA and COKE. Then at each iteration k, each agent i first combines information from its neighbors, i.e., $\boldsymbol{\theta}_j, \forall j \in \mathcal{N}_i$ with its own parameter $\boldsymbol{\theta}_i$ by aggregation. Then, it updates its own parameter $\boldsymbol{\theta}_i$ using the gradient descent method with the aggregated information [85]. The cost function for agent i is given in (2.15). Note that this method has not been formally proposed in existing works for RF-based decentralized kernel learning with batch-form data, but we introduced it here only for comparison purpose. An online version that deals with streaming data is available in [58]. The batch version of CTA introduced here is expected to converge faster than the online version.

DKLA. Algorithm 1 proposed in Section 2.3.2 where ADMM is applied and the communication among agents happen at every iteration without being censored.

The performance of all algorithms is evaluated using both synthetic and real-world

datasets, where the entries of data samples are normalized to lie in $[0, 1]$ and each agent uses 70% of its data for training and the rest for testing. The learning performance at each iteration is evaluated using mean-squared-error (MSE) given by $\text{MSE}(k) = \frac{1}{T}\sum_{i=1}^{N}\sum_{t=1}^{T_i}(y_{i,t} - (\boldsymbol{\theta}_i^k)^\top \boldsymbol{\phi}_L(\mathbf{x}_{i,t}))^2$. The decision variable $\boldsymbol{\theta}_i$ for CTA is initialized as $\boldsymbol{\theta}_i^0 = \mathbf{0}, \forall i$, as that in DKLA and COKE. For COKE, it should be noted that the design of the censoring function is crucial. For the censoring thresholds adopted in Theorem 2, choosing larger v and μ to design the censoring thresholds leads to less communication per iteration but may result in performance degradation. For all simulations, the kernel bandwidth is fine-tuned for each dataset individually via cross-validation. The parameters of the censoring function are tuned to achieve the best learning performance at nearly no performance loss.

2.5.1 Synthetic dataset

In this setup, the connected graph is randomly generated with $N = 20$ nodes and 95 edges. The probability of attachment per node equals to 0.3, i.e., any pair of two nodes are connected with a probability of 0.3. Each agent has $T_i \in (4000, 6000)$ data pairs generated following the model $y_{i,t} = \sum_{m=1}^{50} b_m \kappa(\mathbf{c}_m, \mathbf{x}_{i,t}) + e_{i,t}$, where b_m are uniformly drawn from $[0, 1]$, $\mathbf{c}_m \sim \mathbf{N}(\mathbf{0}, \mathbf{I}_5)$, $\mathbf{x}_{i,t} \sim \mathbf{N}(\mathbf{0}, \mathbf{I}_5)$, and $e_{i,t} \sim \mathbf{N}(0, 0.1)$. The kernel κ in the model is Gaussian with a bandwidth $\sigma = 5$.

2.5.2 Real datasets

To further evaluate our algorithms, the following popular real-world datasets from UCI machine learning repository are chosen [86].

Tom's hardware. This dataset contains $T = 11000$ samples with $\mathbf{x}_t \in \mathbb{R}^{96}$ whose features include the number of created discussions and authors interacting on a topic and $y_t \in \mathbb{R}$ representing the average number of displays to a visitor about that topic [87].

Twitter. This dataset consists of $T = 13800$ samples with $\mathbf{x}_t \in \mathbb{R}^{77}$ being a feature vector reflecting the number of new interactive authors and the length of discussions on a given

topic, etc., and $y_t \in \mathbb{R}$ representing the average number of active discussion on a certain topic. The learning task is to predict the popularity of these topics. We also include a larger Twitter dataset for testing which has $T = 98704$ samples [87].

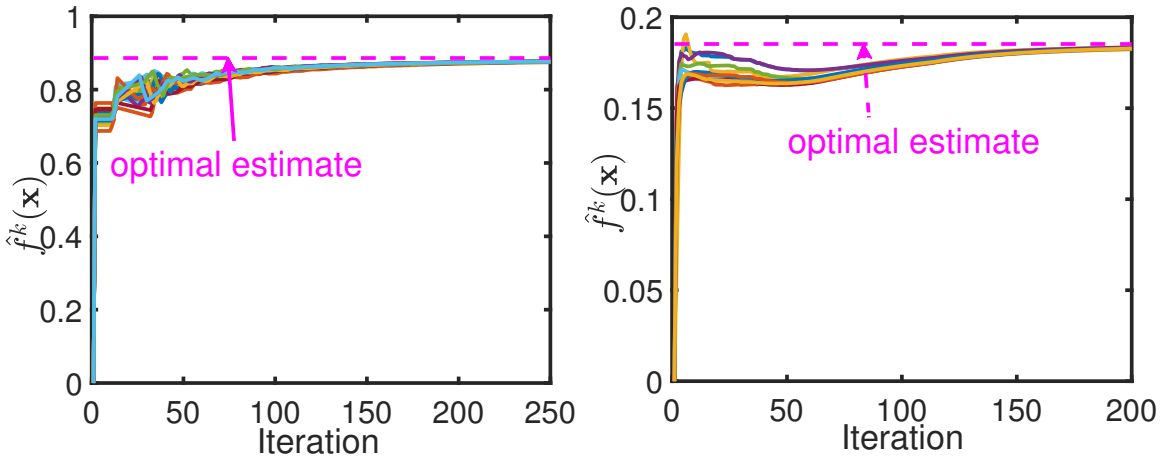
Energy. This dataset contains $T = 19735$ samples with $\mathbf{x}_t \in \mathbb{R}^{28}$ describing the humidity and temperature in different areas of the houses, pressure, wind speed and viability outside, while y_t denotes the total energy consumption in the house [88].

Air quality. This dataset contains dataset collects $T = 9358$ samples measured by a gas multi-sensor device in an Italian city, where $\mathbf{x}_t \in \mathbb{R}^{13}$ represents the hourly concentration of CO, NOx, NO2, etc, while y_t denotes the concentration of polluting chemicals in the air [89].

2.5.3 Parameter setting and performance analysis

For synthetic data, we adopt a Gaussian kernel with a bandwidth $\sigma = 1$ for training and use $L = 100$ random features for kernel approximation. Note that the chosen σ differs from that of the actual data model. The censoring thresholds are $h(k) = 0.95^k$, the regularization parameter λ and stepsize ρ of DKLA and COKE are set to be 5×10^{-5} and 10^{-2}, respectively. The stepsize of CTA is set to be $\eta = 0.99$, which is tuned to achieve the same level of learning performance as COKE and DKLA at its fastest speed.

To show the performance of all algorithms on real datasets concisely and comprehensively, we present the experimental results on the Twitter dataset with $T = 13800$ samples by figures and record the experimental results on the remaining datasets by tables. For the Twitter dataset with $T = 13800$ samples, we randomly split it into 10 mini-batches each with $T_i \in (1200, 1400)$ data pairs while $\sum_{i=1}^{10} T_i = T$. The 10 mini-batches are distributed to 10 agents connected by a random network with 28 edges. We use 100 random features to approximate a Gaussian kernel with a bandwidth $\sigma = 1$ during the training process. The parameters λ and ρ are set to be 10^{-3} and 10^{-2}, respectively. The censoring thresholds are $h(k) = 0.97^k$. The stepsize of CTA is set to be $\eta = 0.99$ to balance the learning performance and the convergence speed.



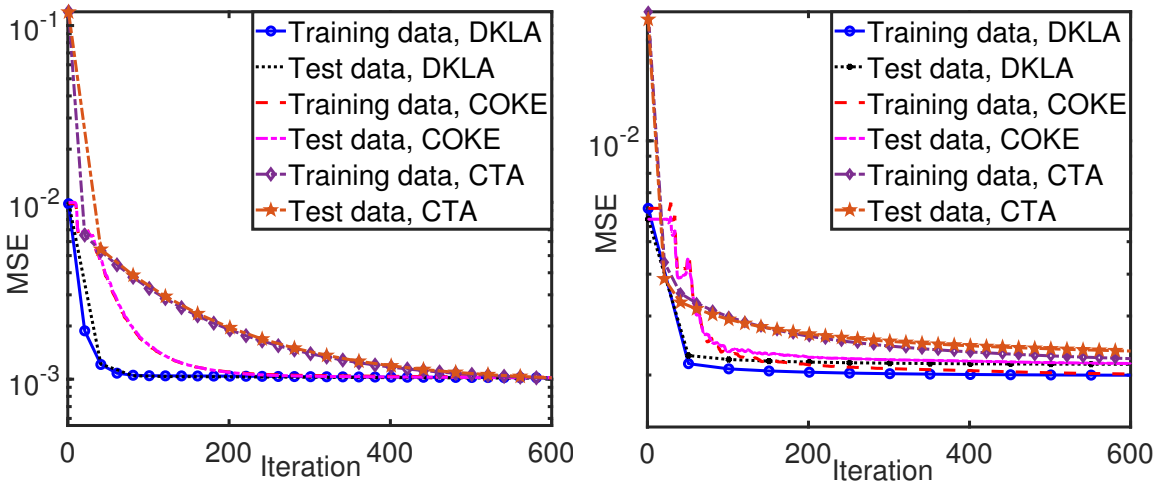
(a) Synthetic data.

(b) Twitter data.

Figure 2.1: Functional convergence via COKE for synthetic data (left) and the real dataset (right). The learned functionals of all distributed agents converge to the optimal estimate where data are assumed to be centrally available.

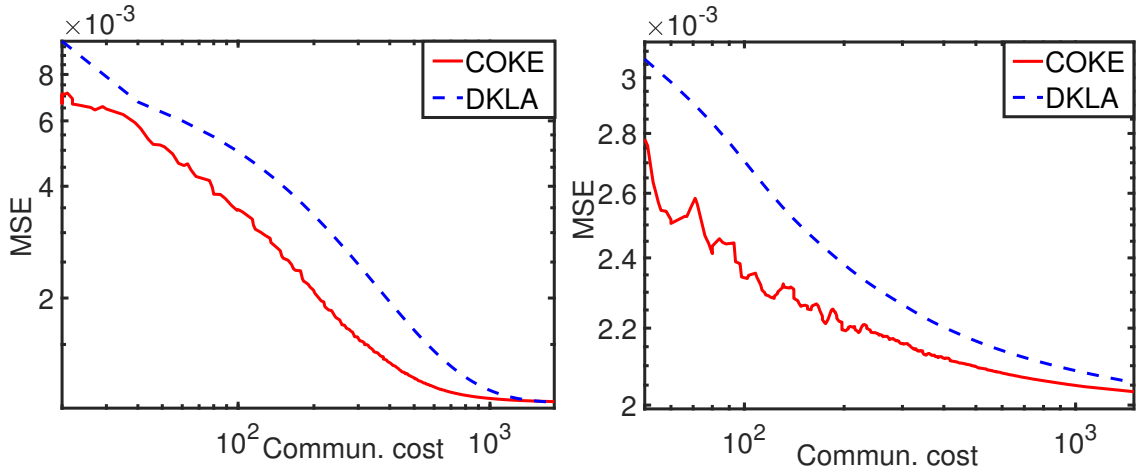
In Figure 2.1, we show that the individually learned functional at each agent via COKE reaches consensus to the optimal estimate for both synthetic and real datasets. In Figure 2.2, we compare the MSE performance of COKE, DKLA, and CTA. Both figures show that COKE converges slower than DKLA due to the communications skipped by the censoring step. However, the learning performance of COKE eventually is the same as DKLA. For the diffusion-based CTA algorithm, it converges the slowest. In Figure 2.3, we show the MSE performance versus the communication cost (in terms of the number of transmissions). As CTA converges the slowest and communicates all the time, its communication cost is much higher than that of DKLA, and thus we do not include it in Figure 2.3 but rather focus on the communication-saving of COKE over DKLA. We can see that to achieve the same level of learning performance, COKE requires much less communication cost than DKLA. Both the synthetic data and the real dataset show communication saving of around 50% in Figure 2.3 at an acceptable learning accuracy, which corroborate the communication efficiency of COKE.

The performance of all three algorithms on the rest four datasets is listed in Table 2.1 -



(a) Synthetic data. (b) Twitter data.

Figure 2.2: MSE performance for synthetic data (left) and the real dataset (right). ADMM-based algorithms (COKE and DKLA) converge faster than the diffusion-based algorithm (CTA) for both synthetic data (left) and the real dataset (right). Furthermore, DKLA and COKE achieve better learning performance than CTA in terms of MSE on the real dataset.



(a) Synthetic data. (b) Twitter data.

Figure 2.3: MSE performance versus communication cost for synthetic data (left) and the real dataset (right). Compared with DKLA, COKE achieves around 50% communication saving at an acceptable MSE performance for both synthetic data (left) and the real dataset (right).

Table 2.1: MSE performance on the Twitter dataset (large), $\sigma = 1$, $L = 100$, $\lambda = 10^{-3}$, stepsize $\eta = 0.99$ for CTA, stepsize $\rho = 10^{-2}$ for DKLA and COKE, censoring thresholds $h(k) = 0.5 \times 0.98^k$. DKLA and COKE achieve better MSE performance than CTA while COKE requires the least communication resource than DKLA.

	Training error (MSE (10^{-3})) / Commun. cost			Test error (MSE(10^{-3}))		
Iteration	CTA	DKLA	COKE	CTA	DKLA	COKE
$k = 50$	3.9/500	2.4/500	4.5/**13**	4.0	2.6	4.2
$k = 100$	3.3/1000	2.4/1000	2.6/**100**	3.4	2.6	2.8
$k = 200$	3.0/2000	2.3/2000	2.4/**298**	3.2	2.5	2.6
$k = 500$	2.7/5000	2.3/5000	**2.3/902**	2.9	2.5	**2.5**
$k = 1000$	2.5/10000	2.2/10000	**2.2/4648**	2.7	2.5	**2.5**
$k = 1500$	2.5/15000	2.2/15000	**2.2/9648**	2.7	2.5	**2.5**
$k = 2000$	2.4/20000	2.2/20000	**2.2/14648**	2.6	2.5	**2.5**

2.6. All results show that COKE saves much communication (almost 50%) within a negligible learning gap from DKLA, and both DKLA and COKE require much less communication resources than CTA. For example, the number of transmissions required to reach a training estimation error of 2.3×10^{-3} on Twitter dataset by COKE is 577, which is only 53% of that required by DKLA to reach the same level of learning performance. For Tom's hardware dataset, COKE requires 361 total transmissions to reach a learning error of 9.95×10^{-4}, which is 56.4% of that required by DKLA and 0.02% of that required by CTA. Note that much of the censoring occurs at the beginning update iterations. While at the later stage, COKE nearly transmits all parameters at every iteration since the censoring thresholds are smaller than the difference between two consecutive updates.

2.6 Concluding Remarks

This chapter studies the decentralized kernel learning problem under privacy concern and communication constraints for multi-agent systems. Leveraging the random feature mapping, we convert the non-parametric kernel learning problem into a parametric one in the RF space and solve it under the consensus optimization framework by the alternating direction method of multipliers. A censoring strategy is applied to conserve communication

Table 2.2: MSE performance on the Tom's hardware dataset, $\sigma = 1$, $L = 100$, $\lambda = 10^{-2}$, stepsize $\eta = 0.99$ for CTA, stepsize $\rho = 10^{-2}$ for DKLA and COKE, censoring thresholds $h(k) = 0.5 \times 0.95^k$. DKLA and COKE achieve better MSE performance than CTA while COKE requires the least communication resource than DKLA.

	Training error (MSE (10^{-4})) / Commun. cost			Test error (MSE(10^{-4}))		
Iteration	CTA	DKLA	COKE	CTA	DKLA	COKE
$k = 50$	20.02/500	10.01/500	17.40/**10**	20.16	11.20	18.82
$k = 100$	16.6/1000	9.91/1000	10.67/**112**	17.09	11.10	11.86
$k = 200$	13.68/2000	9.90/2000	9.97/**331**	14.58	11.10	11.15
$k = 500$	11.19/5000	9.90/5000	**9.90/1114**	12.35	11.10	**11.10**
$k = 1000$	10.27/10000	9.90/10000	**9.90/5600**	11.47	11.10	**11.10**
$k = 1500$	10.01/15000	9.90/15000	**9.90/10600**	11.22	11.10	**11.10**
$k = 2000$	9.92/20000	9.90/20000	**9.90/15600**	11.13	11.10	**11.10**

Table 2.3: MSE performance (training error) versus communication cost on the Twitter dataset (large) and the Tom's hardware dataset. For both datasets, COKE saves around 50% communication resource than DKLA to achieve the same level of learning performance.

Twitter dataset (large)				Tom's hardware			
MSE (10^{-3})	Commun. cost			MSE (10^{-4})	Commun. cost		
	CTA	DKLA	COKE		CTA	DKLA	COKE
5	360	20	**10**	18	680	20	**3**
4	480	30	**10**	16	1020	30	**22**
3	1860	60	**48**	14	1610	60	**28**
2.8	3250	100	**55**	12	2880	110	**51**
2.6	6120	180	**100**	10	7950	250	**128**
2.3	-	1080	**577**	9.95	17620	640	**361**
2.2	-	5660	**4428**	9.90	-	1550	**984**

Table 2.4: MSE performance on the Energy dataset, $\sigma = 0.1$, $L = 100$, $\lambda = 10^{-3}$, stepsize $\eta = 0.99$ for CTA, $\rho = 10^{-2}$ for DKLA and COKE, censoring thresholds $h(k) = 0.5 \times 0.98^k$. DKLA and COKE achieve better MSE performance than CTA while COKE requires the least communication resource.

	Training error (MSE (10^{-3})) / Commun. cost			Test error (MSE(10^{-3}))		
Iteration	CTA	DKLA	COKE	CTA	DKLA	COKE
$k = 50$	25.65/500	22.52/500	25.22/**0**	26.45	22.97	26.02
$k = 100$	24.88/1000	22.12/1000	23.65/**57**	25.57	22.50	24.2
$k = 200$	24.17/2000	21.81/2000	22.57/**254**	24.77	22.15	23.02
$k = 500$	23.40/5000	21.55/5000	21.88/**987**	23.92	21.86	22.22
$k = 1000$	22.84/10000	21.48/10000	21.51/**5752**	23.31	21.79	21.82
$k = 1500$	22.54/15000	21.47/15000	**21.47/10752**	22.97	21.78	**21.78**
$k = 2000$	22.35/20000	21.47/20000	**21.47/15752**	22.75	21.78	**21.78**

Table 2.5: MSE performance on the Air quality dataset, $\sigma = 0.1$, $L = 200$, $\lambda = 10^{-5}$, stepsize $\eta = 0.99$ for CTA, $\rho = 10^{-2}$ for DKLA and COKE, censoring thresholds $h(k) = 0.9 \times 0.97^k$. DKLA and COKE achieve better MSE performance than CTA while COKE requires the least communication resource than DKLA.

	Training error (MSE (10^{-3})) / Commun. cost			Test error (MSE(10^{-3}))		
Iteration	CTA	DKLA	COKE	CTA	DKLA	COKE
$k = 50$	6.4/500	1.8/500	3.7/**72**	6.7	2.1	4.0
$k = 100$	4.5/1000	1.6/1000	2.2/**172**	4.8	1.8	2.5
$k = 200$	3.2/2000	1.4/2000	1.7/**384**	3.5	1.7	2.0
$k = 500$	2.2/5000	1.3/5000	**1.3/2263**	2.5	1.6	**1.6**
$k = 1000$	1.7/10000	1.2/10000	**1.2/7263**	2.0	1.6	**1.6**
$k = 1500$	1.6/15000	1.2/15000	**1.2/12263**	1.8	1.6	**1.6**
$k = 2000$	1.5/20000	1.2/20000	**1.2/17263**	1.8	1.6	**1.6**

Table 2.6: MSE performance (training error) versus communication cost on the Energy dataset and the Air quality dataset. For both datasets, COKE saves around 45%-55% communication resource than DKLA to achieve the same level of learning performance.

Energy				Air quality			
MSE (10^{-3})	Commun. cost			MSE (10^{-3})	Commun. cost		
	CTA	DKLA	COKE		CTA	DKLA	COKE
25	860	20	**11**	5.0	810	60	**49**
24	2290	70	**48**	3.0	2290	180	**81**
23.5	4160	140	**76**	2.0	6010	360	**211**
23	7690	250	**134**	1.8	8160	490	**285**
22.5	14750	480	**258**	1.6	12300	750	**424**
22	-	1160	**652**	1.5	16190	1010	**586**
21.5	-	4950	**4062**	1.2	-	5990	**5383**

resources. Through both theoretical analysis and simulations, we establish that the proposed algorithms not only achieve linear convergence rate but also exhibit effective generalization performance. Thanks to the fixed-size parametric learning model, the proposed algorithms circumvent the curse of dimensionality problem and do not involve raw data exchange among agents. Hence, they can be applied in distributed learning that involve big-data and offer some level of data privacy protection.

2.7 Appendix

2.7.1 Proof of Theorem 1 and Theorem 2

Proof. As discussed in Section 2.3.2, solving the decentralized kernel learning problem in the RF space (2.17) is equivalent to solving the problem (2.16). From (2.14), it is evident that the convergence of the optimal functional f in (2.16) hinges on the convergence of the decision variables $\boldsymbol{\theta}$ in the RF space. Since in the RF space, the decision variables are data-independent, the convergence proof of DKLA boils down to proving the convergence of a convex optimization problem solved by ADMM. However, the convergence proof of COKE is nontrivial because of the error caused by the outdated information introduced by the communication censoring strategy. Our proof for both theorems consists of two steps. The first step is to show linear convergence of decision parameters $\boldsymbol{\theta}$ for DKLA via Theorem 4 and for COKE via Theorem 5 below, which are derived straightforwardly from [27] and [59], respectively. The second step is to show how the convergence of $\boldsymbol{\theta}$ translates to the convergence of the learned functional, which are the same for both algorithms.

Compared to [27] and [59] that deal with general optimization problems for parametric learning, this work focuses on the specific decentralized kernel learning problem which is more challenging in both solution development and theoretical analysis. By leveraging the RF mapping technique, we successfully develop the DKLA algorithm and the COKE algorithm. Noticeably, a direct application of ADMM as in [27] on decentralized kernel learning is infeasible without raw data exchanges. Moreover, we analyze the convergence of the nonlinear functional to be learned and the generalization performance of kernel learning in the decentralized setting. The analysis is built on the work of [27] and [59] but goes further, and it is only attainable because of the adoption of the RF mapping.

For both algorithms, the linear convergence of decision variables in the RF space is based on matrix reformulation of (2.17). Define $\boldsymbol{\Theta}^\star := [\boldsymbol{\theta}^\star, \boldsymbol{\theta}^\star, \ldots, \boldsymbol{\theta}^\star]^\top \in \mathbb{R}^{N\times L}$ and $\boldsymbol{Z}^\star := [\boldsymbol{z}^\star, \boldsymbol{z}^\star, \ldots, \boldsymbol{z}^\star]^\top \in \mathbb{R}^{N\times L}$ be the optimal primal variables, and $\boldsymbol{\beta}^\star$ be the optimal dual variable. Then, for DKLA, Theorem 4 states that $\{\boldsymbol{\Theta}^k\}$ ($\boldsymbol{\Theta}^k := [\boldsymbol{\theta}_1^k, \boldsymbol{\theta}_2^k, \ldots, \boldsymbol{\theta}_N^k]^\top \in \mathbb{R}^{N\times L}$)

is R-linear convergent to the optimal $\boldsymbol{\Theta}^\star$. For detailed proof, see [27].

Theorem 4. ***[Linear convergence of decision variables in DKLA]*** *For the optimization problem* (2.16)*, initialize the dual variables as* $\boldsymbol{\gamma}_i^0 = \mathbf{0}$, $\forall i$*, with Assumptions 1 and 2, then* $\{\boldsymbol{\Theta}^k\}$ *is R-linearly convergent to the optimal* $\boldsymbol{\Theta}^\star$ *when* k *goes to infinity following from*

$$\|\boldsymbol{\Theta}^k - \boldsymbol{\Theta}^\star\|_F^2 \leq \frac{1}{m_{\hat{R}}}\left[\rho\|\boldsymbol{Z}^{k-1} - \boldsymbol{Z}^\star\|_F^2 + \frac{1}{\rho}\|\boldsymbol{\beta}^{k-1} - \boldsymbol{\beta}^\star\|_F^2\right], \tag{2.29}$$

where $\{(\boldsymbol{Z}^k, \boldsymbol{\beta}^k)\}$ *is Q-linearly convergent to its optimal* $\{(\boldsymbol{Z}^\star, \boldsymbol{\beta}^\star)\}$*:*

$$\rho\|\boldsymbol{Z}^k - \boldsymbol{Z}^\star\|_F^2 + \frac{1}{\rho}\|\boldsymbol{\beta}^k - \boldsymbol{\beta}^\star\|_F^2 \leq \frac{1}{1+\delta_d}\left[\rho\|\boldsymbol{Z}^{k-1} - \boldsymbol{Z}^\star\|_F^2 + \frac{1}{\rho}\|\boldsymbol{\beta}^{k-1} - \boldsymbol{\beta}^\star\|_F^2\right] \tag{2.30}$$

with

$$\delta_d = \min\left\{\frac{(\nu-1)\tilde{\sigma}_{\min}^2(\mathbf{S}_-)}{\nu\tilde{\sigma}_{\max}^2(\mathbf{S}_+)}, \frac{m_{\hat{R}}}{\frac{\rho}{4}\tilde{\sigma}_{\max}^2(\mathbf{S}_+) + \frac{\nu}{\rho}M_{\hat{R}}^2\tilde{\sigma}_{\min}^2(\mathbf{S}_-)}\right\},$$

where $\nu > 1$ *is an arbitrary constant,* $\tilde{\sigma}_{\max}(\mathbf{S}_+)$ *is the maximum singular value of the unsigned incidence matrix* $\mathbf{S}_+$ *of the network, and* $\tilde{\sigma}_{\min}^2(\mathbf{S}_-)$ *is the minimum non-zero singular value of the signed incidence matrix* $\mathbf{S}_-$ *of the network,* $m_{\hat{R}}$ *and* $M_{\hat{R}}$ *are the minimum strong convexity constant of the local cost functions and the maximum Lipschitz constant of the local gradients, respectively. The Q-linear convergence rate of* $\{(\boldsymbol{Z}^k, \boldsymbol{\beta}^k)\}$ *to* $\{(\boldsymbol{Z}^\star, \boldsymbol{\beta}^\star)\}$ *satisfies*

$$r_c \leq \sqrt{\frac{1}{1+\delta_d}}. \tag{2.31}$$

To achieve linear convergence of decision variables in COKE, choosing appropriate censoring functions is crucial. Moreover, the penalty parameter ρ also needs to satisfy certain conditions, see Theorem 5 for details [59].

Theorem 5. ***[Linear convergence of decision variables in COKE]*** *For the optimization problem* (2.16) *with strongly convex local cost functions whose gradients are Lipschitz continuous, initialize the dual variables as* $\boldsymbol{\gamma}_i^0 = \mathbf{0}$, $\forall i$, *set the censoring thresholds to be* $h(k) = v\mu^k$, *with* $v > 0$ *and* $\mu \in (0, 1)$, *and choose the penalty parameter* ρ *such that*

$$0 < \rho < \min\left\{\frac{4m_{\hat{R}}}{\eta_1}, \frac{(\nu-1)\tilde{\sigma}^2_{\min}(\mathbf{S}_-)}{\nu\eta_3\tilde{\sigma}^2_{\max}(\mathbf{S}_+)}, \left(\frac{\eta_1}{4} + \frac{\eta_2\tilde{\sigma}^2_{\max}(\mathbf{S}_+)}{8}\right)^{-1}\left(m_{\hat{R}} - \frac{\eta_3\nu M^2_{\hat{R}}}{\tilde{\sigma}^2_{\min}(\mathbf{S}_-)}\right)\right\}, \tag{2.32}$$

where $\eta_1 > 0$, $\eta_2 > 0$, $\eta_3 > 0$ *and* $\nu > 1$ *are arbitrary constants,* $m_{\hat{R}}$ *and* $M_{\hat{R}}$ *are the minimum strong convexity constant of the local cost functions and the maximum Lipschitz constant of the local gradients, respectively.* $\tilde{\sigma}_{\max}(\mathbf{S}_+)$ *and* $\tilde{\sigma}^2_{\min}(\mathbf{S}_-)$ *are the maximum singular value of the unsigned incidence matrix* $\mathbf{S}_+$ *and the minimum non-zero singular value of the signed incidence matrix* $\mathbf{S}_-$ *of the network, respectively. Then,* $\{\boldsymbol{\Theta}^k\}$ *is R-linearly convergent to the optimal* $\boldsymbol{\Theta}^\star$ *when* k *goes to infinity.*

Remark 3. For the kernel ridge regression problem (2.25), the minimum strong convexity constant of the local cost functions and the maximum Lipschitz constant of the local gradients are $m_{\hat{R}} := \min_i \tilde{\sigma}^2_{\min}(\frac{1}{T_i}\boldsymbol{\Phi}_L^i(\boldsymbol{\Phi}_L^i)^\top + \frac{2\lambda}{N}\mathbf{I})$ and $M_{\hat{R}} := \max_i \tilde{\sigma}^2_{\max}(\frac{1}{T_i}\boldsymbol{\Phi}_L^i(\boldsymbol{\Phi}_L^i)^\top + \frac{2\lambda}{N}\mathbf{I})$, respectively.

With the convergence of decision variables in the RF space given in Theorem 4 and Theorem 5, the second step is to prove the linear convergence of the learned functional $\hat{f}_{\boldsymbol{\theta}_i^k}(\mathbf{x})$ to the optimal $\hat{f}_{\boldsymbol{\theta}^\star}(\mathbf{x})$, which is straightforward for both algorithms.

Denote $\hat{\boldsymbol{f}}_{\boldsymbol{\Theta}^k}(\mathbf{x}) = [\hat{f}_{\boldsymbol{\theta}_1^k}(\mathbf{x}), \ldots, \hat{f}_{\boldsymbol{\theta}_N^k}(\mathbf{x})]^\top = \boldsymbol{\Theta}^k\boldsymbol{\phi}_L(\mathbf{x})$ and $\hat{\boldsymbol{f}}_{\boldsymbol{\Theta}^\star}(\mathbf{x}) = [\hat{f}_{\boldsymbol{\theta}^\star}(\mathbf{x}), \ldots, \hat{f}_{\boldsymbol{\theta}^\star}(\mathbf{x})]^\top = \boldsymbol{\Theta}^\star\boldsymbol{\phi}_L(\mathbf{x})$, then we have

$$\begin{aligned}\|\hat{\boldsymbol{f}}_{\boldsymbol{\Theta}^k}(\mathbf{x}) - \hat{\boldsymbol{f}}_{\boldsymbol{\Theta}^\star}(\mathbf{x})\|_2 &= \|\boldsymbol{\Theta}^k\boldsymbol{\phi}_L(\mathbf{x}) - \boldsymbol{\Theta}^\star\boldsymbol{\phi}_L(\mathbf{x})\|_2 \\ &\leq \|\boldsymbol{\Theta}^k - \boldsymbol{\Theta}^\star\|_2\|\boldsymbol{\phi}_L(\mathbf{x})\|_2 \\ &\leq \|\boldsymbol{\Theta}^k - \boldsymbol{\Theta}^\star\|_2,\end{aligned} \tag{2.33}$$

where the second inequality comes from the fact that $\|\boldsymbol{\phi}_L(\mathbf{x})\|_2 \leq 1$ with the adopted RF mapping.

For DKLA, we have

$$\|\hat{\boldsymbol{f}}_{\boldsymbol{\Theta}^k}(\mathbf{x}) - \hat{\boldsymbol{f}}_{\boldsymbol{\Theta}^\star}(\mathbf{x})\|_2 \leq \|\boldsymbol{\Theta}^k - \boldsymbol{\Theta}^\star\|_2 \leq \frac{1}{m_{\hat{R}}}\left[\rho\|\boldsymbol{Z}^{k-1} - \boldsymbol{Z}^\star\|_F^2 + \frac{1}{\rho}\|\boldsymbol{\beta}^{k-1} - \boldsymbol{\beta}^\star\|_F^2\right]. \tag{2.34}$$

Therefore, the Q-linear convergence of $\{\boldsymbol{Z}^k, \boldsymbol{\beta}^k\}$ to the optimal $(\boldsymbol{Z}^\star, \boldsymbol{\beta}^\star)$ translates to the R-linear convergence of $\{\hat{\boldsymbol{f}}_{\boldsymbol{\Theta}^k}(\mathbf{x})\}$. Similarly, the R-linear convergence of $\{\boldsymbol{\Theta}^k\}$ to the optimal $\boldsymbol{\Theta}^\star$ of COKE can be translated from the Q-linear convergence of $\{\boldsymbol{Z}^k, \boldsymbol{\beta}^k\}$ to the optimal $(\boldsymbol{Z}^\star, \boldsymbol{\beta}^\star)$, see [59] for detailed proof.

It is then straightforward to see that the individually learned functionals converge to the optimal one when k goes to infinity, that is, $\forall i \in \mathcal{N}$,

$$\begin{aligned}\lim_{k\to\infty} |\hat{f}_{\boldsymbol{\theta}_i^k}(\mathbf{x}) - \hat{f}_{\boldsymbol{\theta}^\star}(\mathbf{x})| &= \lim_{k\to\infty} |(\boldsymbol{\theta}_i^k)^\top\boldsymbol{\phi}_L(\mathbf{x}) - (\boldsymbol{\theta}^\star)^\top\boldsymbol{\phi}_L(\mathbf{x})| \\ &\leq \lim_{k\to\infty} \|\boldsymbol{\theta}_i^k - \boldsymbol{\theta}^\star\|_2\|\boldsymbol{\phi}_L(\mathbf{x})\|_2 \\ &\leq \lim_{k\to\infty} \|\boldsymbol{\theta}_i^k - \boldsymbol{\theta}^\star\|_2 \\ &= 0,\end{aligned} \tag{2.35}$$

which completes the proof of Theorem 1 and 2. ■

2.7.2 Proof of Theorem 3

Proof. The empirical risk (2.6) to be minimized for the kernel regression problem in the RKHS is

$$\min_{\boldsymbol{\alpha}\in\mathbb{R}^T} \hat{R}(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \hat{R}_i(\boldsymbol{\alpha}) = \sum_{i=1}^{N} \left(\frac{1}{T_i} \|\mathbf{y}_i - \mathbf{K}_i^\top \boldsymbol{\alpha}\|_2^2 + \lambda_i \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} \right), \tag{2.36}$$

where $\mathbf{y}_i = [y_{i,1}, \ldots, y_{i,T_i}]^\top \in \mathbb{R}^{T_i \times 1}$, the matrices $\mathbf{K}_i \in \mathbb{R}^{T \times T_i}$ and $\mathbf{K} \in \mathbb{R}^{T \times T}$ are used to store the similarity of the total data and data from agent i, and the similarity of all data, respectively, with the assumption that all data are available to all agents. The optimal solution is given in closed form by

$$\boldsymbol{\alpha}^\star = (\tilde{\mathbf{K}}^\top \tilde{\mathbf{K}} + \lambda \mathbf{K})^{-1} \tilde{\mathbf{K}} \tilde{\mathbf{y}}, \tag{2.37}$$

where $\tilde{\mathbf{K}} = [\tilde{\mathbf{K}}_1, \ldots, \tilde{\mathbf{K}}_N] \in \mathbb{R}^{T \times T}$ with $\tilde{\mathbf{K}}_i = \frac{1}{\sqrt{T_i}} \mathbf{K}_i$, $\forall i \in \mathcal{N}$, $\tilde{\mathbf{y}} = [\tilde{\mathbf{y}}_1; \ldots; \tilde{\mathbf{y}}_N] \in \mathbb{R}^{T \times 1}$ with $\tilde{\mathbf{y}}_i = \frac{1}{\sqrt{T_i}} \mathbf{y}_i$, $\forall i \in \mathcal{N}$, and $\lambda = \sum_{i=1}^{N} \lambda_i$. Denote the predicted values on the training examples using $\boldsymbol{\alpha}^\star$ as $\mathbf{f}_{\boldsymbol{\alpha}^\star}^i \in \mathbb{R}^{T_i}$ for node i and the overall predictions as $\mathbf{f}_{\boldsymbol{\alpha}^\star} = [\mathbf{f}_{\boldsymbol{\alpha}^\star}^1; \ldots; \mathbf{f}_{\boldsymbol{\alpha}^\star}^N] \in \mathbb{R}^T$. In the corresponding RF space, we can denote the predicted values obtained for node i by $\boldsymbol{\theta}^\star$ in (2.26) as $\mathbf{f}_{\boldsymbol{\theta}^\star}^i \in \mathbb{R}^{T_i}$ and the overall prediction by $\mathbf{f}_{\boldsymbol{\theta}^\star} = [\mathbf{f}_{\boldsymbol{\theta}^\star}^1; \ldots; \mathbf{f}_{\boldsymbol{\theta}^\star}^N] \in \mathbb{R}^T$.

To prove Theorem 3, we start by customizing several lemmas and theorems from the literature, which facilitate proving our main results.

Definition 1. *[90, Definition 2] Let $\{\mathbf{x}_q\}_{q=1}^Q$ be i.i.d samples drawn from the probability distribution $p_{\mathcal{X}}$. Let $\mathcal{F}$ be a class of functions that map $\mathcal{X}$ to $\mathbb{R}$. Define the random variable*

$$\hat{\mathfrak{R}}_Q(\mathcal{F}) := \mathbb{E}_\epsilon \left[\sup_{f\in\mathcal{F}} \left| \frac{2}{Q} \sum_{q=1}^{Q} \epsilon_q f(\mathbf{x}_q) \right| | \mathbf{x}_1, \ldots, \mathbf{x}_Q \right], \tag{2.38}$$

where $\{\epsilon_q\}_{q=1}^Q$ are i.i.d. $\{\pm 1\}$-valued random variables with $\mathbb{P}(\epsilon_q = 1) = \mathbb{P}(\epsilon_q = -1) = \frac{1}{2}$.

Then, the Rademacher complexity of $\mathcal{F}$ is defined as

$$\mathfrak{R}_Q(\mathcal{F}) := \mathbb{E}\left[\hat{\mathfrak{R}}_Q(\mathcal{F})\right]. \tag{2.39}$$

Rademacher complexity is adopted in machine learning and theory of computation to measure the richness of a class of real-valued functions with respect to a probability distribution. Here we adopt it to measure the richness of functions defined in the RKHS induced by the positive definite kernel κ with respect to the sample distribution p.

Lemma 1. *[90, Lemma 22] Let $\mathcal{H}$ be a RKHS associated with a positive definite kernel κ that maps $\mathcal{X}$ to $\mathbb{R}$. Then, we have $\hat{\mathfrak{R}}_Q(\mathcal{H}) \leq \frac{2}{Q}\sqrt{Tr(\mathbf{K})}$, where $\mathbf{K}$ is the kernel matrix for kernel κ over the i.i.d. sample set $\{\mathbf{x}_q\}_{q=1}^Q$. Correspondingly, the Rademacher complexity satisfies $\mathfrak{R}_Q(\mathcal{H}) \leq \frac{2}{Q}\mathbb{E}\left[\sqrt{Tr(\mathbf{K})}\right]$.*

The next theorems state that the generalization performance of a particular estimator in $\mathcal{H}$ not only depends on the number of data points, but also depends on the complexity of $\mathcal{H}$.

Theorem 6. *[90, Theorem 8, Theorem 12] Let $\{\mathbf{x}_q, y_q\}_{q=1}^Q$ be i.i.d samples drawn from the distribution p defined on $\mathcal{X} \times \mathcal{Y}$. Assume the loss function $\ell : \mathcal{Y} \times \mathbb{R} \to [0,1]$ is Lipschitz continuous with a Lipschitz constant M_ℓ. Define the expected risk for all $f \in \mathcal{H}$ be $\mathcal{E}(f) = \mathbb{E}_p[\ell(f(\mathbf{x}), y)]$, and its corresponding empirical risk be $\hat{\mathcal{E}}(f) = \frac{1}{Q}\sum_{q=1}^Q \ell(y_q, f(\mathbf{x}_q))$. Then, for $\delta_p \in (0,1)$, with probability at least $1-\delta_p$, every $f \in \mathcal{H}$ satisfies*

$$\mathcal{E}(f) \leq \hat{\mathcal{E}}(f) + \mathfrak{R}_Q(\tilde{\ell} \circ \mathcal{H}) + \sqrt{\frac{8\log(2/\delta_p)}{Q}}, \tag{2.40}$$

where $\tilde{\ell} \circ \mathcal{H} = \{(\mathbf{x}, y) \to \ell(y, f(\mathbf{x})) - \ell(y, 0) | f \in \mathcal{H}\}$.

Theorem 7. *[90, Theorem 12] If $\ell : \mathcal{Y} \times \mathbb{R} \to [0,1]$ is Lipschitz with constant M_ℓ and*

satisfies $\ell(0) = 0$, *then* $\mathfrak{R}_Q(\tilde{\ell} \circ \mathcal{H}) \leq 2M_\ell \mathfrak{R}_Q(\mathcal{H})$.

Lemma 2. *[91, Modified Proposition 1] For the RKHS induced by the kernel* κ *with expression* (2.10), *define* $\hat{\mathcal{H}}^k := \{\hat{f}^k : \hat{f}^k = (\boldsymbol{\theta}^k)^\top \boldsymbol{\phi}_L(\mathbf{x}) = \sum_{l=1}^L \theta_l^k \phi(\mathbf{x}, \boldsymbol{\omega}_l)\}$, *then we have* $\forall \hat{f}^k \in \hat{\mathcal{H}}^k, \|\hat{f}^k\|_{\hat{\mathcal{H}}^k}^2 \leq \|\boldsymbol{\theta}^k\|_2^2$, *where* $\hat{\mathcal{H}}^k$ *is the RKHS of functions* $\hat{f}^k$ *at the* k*-th step. The kernel that induces* $\hat{\mathcal{H}}^k$ *is the approximated kernel* $\hat{\kappa}_L$ *defined in* (2.11).

Lemma 3. *[91, Lemma 6] For the decentralized kernel regression problem defined in Section 2.2, let* $\mathbf{f}_{\boldsymbol{\alpha}^\star}$, $\mathbf{f}_{\boldsymbol{\theta}^\star}$ *be the predictions obtained by* (2.37) *and* (2.26), *respectively. Then, we have*

$$\langle \mathbf{y} - \mathbf{f}_{\boldsymbol{\alpha}^\star}, \mathbf{f}_{\boldsymbol{\theta}^\star} - \mathbf{f}_{\boldsymbol{\alpha}^\star} \rangle = 0. \tag{2.41}$$

Theorem 8. *[91, Modified Theorem 5] For the decentralized kernel regression problem defined in Section 2.2, let* $\lambda_{\mathbf{K}}$ *be the largest eigenvalue of the kernel matrix* $\mathbf{K}$, *and choose the regularization parameter* $\lambda < \lambda_{\mathbf{K}}/T$ *so as to control overfitting. Then, for all* $\delta_p \in (0, 1)$ *and* $\|f\|_{\mathcal{H}} \leq 1$, *if the number of random features* L *satisfies*

$$L \geq \frac{1}{\lambda}\left(\frac{1}{\epsilon^2} + \frac{2}{3\epsilon}\right) \log \frac{16 d_{\mathbf{K}}^{\lambda}}{\delta_p},$$

then with probability at least $1 - \delta_p$, *the following equation holds*

$$\sup_{\|f\|_{\mathcal{H}} \leq 1} \inf_{\|\boldsymbol{\theta}\| \leq \sqrt{2/L}} \frac{1}{T} \|\mathbf{f}_{\mathbf{x}} - \mathbf{f}_{\boldsymbol{\theta}^\star}\|_2^2 \leq 2\lambda, \tag{2.42}$$

where $\mathbf{f}_{\mathbf{x}} \in \mathbb{R}^T$ *is the predictions evaluated by* $f_{\mathcal{H}}$ *on all samples and* $\epsilon \in (0, 1)$.

With the above lemmas and theorems, we are ready to prove Theorem 3, which relies

on the following decomposition:

$$\mathcal{E}(\hat{f}^k) - \mathcal{E}(f_{\mathcal{H}}) = \underbrace{\mathcal{E}(\hat{f}^k) - \hat{\mathcal{E}}(\hat{f}^k)}_{\text{(1) estimation error}} + \underbrace{\hat{\mathcal{E}}(\hat{f}^k) - \hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\theta}^\star})}_{\text{(2) convergence error}} + \underbrace{\hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\theta}^\star}) - \hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\alpha}^\star})}_{\text{(3) approximation error of RF mapping}}$$
$$+ \underbrace{\hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\alpha}^\star}) - \mathcal{E}(\hat{f}_{\boldsymbol{\alpha}^\star})}_{\text{(4) estimation error}} + \underbrace{\mathcal{E}(\hat{f}_{\boldsymbol{\alpha}^\star}) - \mathcal{E}(f_{\mathcal{H}})}_{\text{(5) approximation error of kernel representation}}, \tag{2.43}$$

where $\mathcal{E}(\hat{f}^k)$, $\hat{\mathcal{E}}(\hat{f}^k)$, $\mathcal{E}(\hat{f}_{\boldsymbol{\theta}^\star})$, $\hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\theta}^\star})$, $\hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\alpha}^\star})$, $\mathcal{E}(\hat{f}_{\boldsymbol{\alpha}^\star})$ are defined as follows for the kernel regression problem:

$$\mathcal{E}(\hat{f}^k) := \sum_{i=1}^{N} \mathcal{E}_i(\hat{f}_{\boldsymbol{\theta}_i^k}) = \sum_{i=1}^{N} \mathbb{E}_p[(y - (\boldsymbol{\theta}_i^k)^\top \boldsymbol{\phi}_L(\mathbf{x}))^2] := \mathbb{E}_p[\|\mathbf{y}_N - \boldsymbol{\Phi}_N \tilde{\boldsymbol{\Theta}}^k\|_2^2],$$

$$\hat{\mathcal{E}}(\hat{f}^k) := \sum_{i=1}^{N} \hat{\mathcal{E}}_i(\hat{f}_{\boldsymbol{\theta}_i^k}) = \sum_{i=1}^{N} \frac{1}{T_i} \sum_{t=1}^{T_i} \|\mathbf{y}_i - (\boldsymbol{\Phi}_L^i)^\top \boldsymbol{\theta}_i^k\|_2^2 = \sum_{i=1}^{N} \|\tilde{\mathbf{y}}_i - (\tilde{\boldsymbol{\Phi}}_L^i)^\top \boldsymbol{\theta}_i^k\|_2^2 = \|\tilde{\mathbf{y}} - \tilde{\boldsymbol{\Phi}}_B \tilde{\boldsymbol{\Theta}}^k\|_2^2,$$

$$\hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\theta}^\star}) := \sum_{i=1}^{N} \hat{\mathcal{E}}_i(\hat{f}_{\boldsymbol{\theta}^\star}) = \sum_{i=1}^{N} \frac{1}{T_i} \sum_{t=1}^{T_i} \|\mathbf{y}_i - (\boldsymbol{\Phi}_L^i)^\top \boldsymbol{\theta}^\star\|_2^2 = \sum_{i=1}^{N} \|\tilde{\mathbf{y}}_i - (\tilde{\boldsymbol{\Phi}}_L^i)^\top \boldsymbol{\theta}^\star\|_2^2 = \|\tilde{\mathbf{y}} - \tilde{\boldsymbol{\Phi}}_B \tilde{\boldsymbol{\Theta}}^\star\|_2^2,$$

$$\mathcal{E}(\hat{f}_{\boldsymbol{\alpha}^\star}) := \sum_{i=1}^{N} \hat{\mathcal{E}}_i(\hat{f}_{\boldsymbol{\alpha}^\star}) = \sum_{i=1}^{N} \mathbb{E}_p[(y - (\boldsymbol{\alpha}^\star)^\top \boldsymbol{\kappa}(\mathbf{x}))^2] := \mathbb{E}_p[(y - (\boldsymbol{\alpha}^\star)^\top \boldsymbol{\kappa}(\mathbf{x}))^2],$$

$$\hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\alpha}^\star}) := \sum_{i=1}^{N} \frac{1}{T_i} \|\mathbf{y}_i - \mathbf{K}_i \boldsymbol{\alpha}^\star\|_2^2 = \sum_{i=1}^{N} \|\tilde{\mathbf{y}}_i - \tilde{\mathbf{K}}_i \boldsymbol{\alpha}^\star\|_2^2,$$

where $\mathbf{y}_N = y\mathbf{1}_N$, $\boldsymbol{\Phi}_N = \begin{bmatrix} \boldsymbol{\phi}_L(\mathbf{x}) & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & \boldsymbol{\phi}_L(\mathbf{x}) \end{bmatrix} \in \mathbb{R}^{N \times NL}$, $\tilde{\boldsymbol{\Phi}}_B = \begin{bmatrix} (\tilde{\boldsymbol{\Phi}}_L^1)^\top & \cdots & \mathbf{0} \\ \vdots & \ddots & \vdots \\ \mathbf{0} & \cdots & (\tilde{\boldsymbol{\Phi}}_L^N)^\top \end{bmatrix} \in$ $\mathbb{R}^{T \times NL}$, $\tilde{\boldsymbol{\Theta}}^k = [\boldsymbol{\theta}_1^k; \ldots; \boldsymbol{\theta}_i^k] \in \mathbb{R}^{NL}$, and $\tilde{\boldsymbol{\Theta}}^\star = [\boldsymbol{\theta}^\star; \ldots; \boldsymbol{\theta}^\star] \in \mathbb{R}^{NL}$.

Then, we upper bound the excessive risk of $\mathcal{E}(\hat{f}^k)$ learned by COKE by upper bounding the decomposed five terms. For term (1), for $\delta_{p_1} \in (0, 1)$, with probability at least $1 - \delta_{p_1}$,

we have

$$\begin{aligned}
\mathcal{E}(\hat{f}^k) - \hat{\mathcal{E}}(\hat{f}^k) &\leq 2M_{\ell_1}\mathfrak{R}_T(\tilde{\ell}_1 \circ \hat{\mathcal{H}}^k) + \sqrt{\frac{8\log(2/\delta_{p_1})}{T}} \\
&\leq 2M_{\ell_1}\mathfrak{R}_T(\hat{\mathcal{H}}^k) + \sqrt{\frac{8\log(2/\delta_{p_1})}{T}} \\
&\leq \frac{4M_{\ell_1}}{T}\mathbb{E}[Tr(\hat{\mathbf{K}})] + \sqrt{\frac{8\log(2/\delta_{p_1})}{T}} \\
&\leq \frac{4M_{\ell_1}}{T}\sqrt{T} + \sqrt{\frac{8\log(2/\delta_{p_1})}{T}} \\
&= \frac{C_1}{\sqrt{T}},
\end{aligned} \tag{2.44}$$

where $C_1 := 4M_{\ell_1} + \sqrt{8\log(2/\delta_{p_1})}$, and M_{ℓ_1} is the Lipschitz constant for loss function $\ell_1(\hat{f}_{\boldsymbol{\theta}_i^k}, y) = ((\boldsymbol{\theta}_i^k)^\top \boldsymbol{\phi}_L(\mathbf{x}) - y)^2$. The first inequality comes from Theorem 6, the second inequality comes from Theorem 7, and the third inequality comes from Lemma 1. For the last inequality, each element in the Gram matrix $\hat{\mathbf{K}} \in \mathbb{R}^{T\times T}$ is given by (2.11), thus $Tr(\hat{\mathbf{K}}) \leq T\|\boldsymbol{\phi}_L(\mathbf{x})\|_2^2 \leq T$ with the adopted RF mapping such that $\|\boldsymbol{\phi}_L(\mathbf{x})\|_2^2 \leq 1$.

Similarly, for term (4), with probability at least $1 - \delta_{p_2}$ for $\delta_{p_2} \in (0, 1)$, the following holds,

$$\begin{aligned}
\hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\alpha}^\star}) - \mathcal{E}(\hat{f}_{\boldsymbol{\alpha}^\star}) &\leq 2M_{\ell_2}\mathfrak{R}_T(\tilde{\ell}_2 \circ \mathcal{H}) + \sqrt{\frac{8\log(2/\delta_{p_2})}{T}} \\
&\leq 2M_{\ell_2}\mathfrak{R}_T(\mathcal{H}) + \sqrt{\frac{8\log(2/\delta_{p_2})}{T}} \\
&\leq \frac{4M_{\ell_2}}{T}\mathbb{E}[Tr(\mathbf{K})] + \sqrt{\frac{8\log(2/\delta_{p_2})}{T}} \\
&\leq \frac{4M_{\ell_2}}{T}\sqrt{T} + \sqrt{\frac{8\log(2/\delta_{p_2})}{T}} \\
&= \frac{C_2}{\sqrt{T}},
\end{aligned} \tag{2.45}$$

where $C_2 := 4M_{\ell_2} + \sqrt{8\log(2/\delta_{p_2})}$, and M_{ℓ_2} is the Lipschitz constant for the loss function $\ell_2(\hat{f}_{\boldsymbol{\alpha}^\star}, y) = ((\boldsymbol{\alpha}^\star)^\top \boldsymbol{\kappa}(\mathbf{x}) - y)^2$.

For term (2), we have

$$\begin{aligned}
\hat{\mathcal{E}}(\hat{f}^k) - \hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\theta}^\star}) &= \|\tilde{\mathbf{y}} - \tilde{\boldsymbol{\Phi}}_B \tilde{\boldsymbol{\Theta}}^k\|_2^2 - \|\tilde{\mathbf{y}} - \tilde{\boldsymbol{\Phi}}_B \tilde{\boldsymbol{\Theta}}^\star\|_2^2 \\
&\leq \nabla \left(\|\tilde{\mathbf{y}} - \tilde{\boldsymbol{\Phi}}_B \tilde{\boldsymbol{\Theta}}^\star\|_2^2 \right) \|\tilde{\boldsymbol{\Theta}}^k - \tilde{\boldsymbol{\Theta}}^\star\|_2 + \frac{M_{\ell_3}}{2} \|\tilde{\boldsymbol{\Theta}}^k - \tilde{\boldsymbol{\Theta}}^\star\|_2 \\
&\leq \left(\|\tilde{\boldsymbol{\Phi}}_B^\top (\tilde{\boldsymbol{\Phi}}_B \tilde{\boldsymbol{\Theta}}^\star - \tilde{\mathbf{y}})\|_2 + \frac{M_{\ell_3}}{2} \right) \|\tilde{\boldsymbol{\Theta}}^k - \tilde{\boldsymbol{\Theta}}^\star\|_2 \\
&= C_3 \|\tilde{\boldsymbol{\Theta}}^k - \tilde{\boldsymbol{\Theta}}^\star\|_2,
\end{aligned} \tag{2.46}$$

where $C_3 := \|\tilde{\boldsymbol{\Phi}}_B^\top (\tilde{\boldsymbol{\Phi}}_B \tilde{\boldsymbol{\Theta}}^\star - \tilde{\mathbf{y}})\|_2 + \frac{M_{\ell_3}}{2}$, and M_{ℓ_3} is the Lipschitz constant of the loss function $\ell_3(\tilde{\mathbf{y}}, \tilde{\boldsymbol{\Theta}}) = \|\tilde{\mathbf{y}} - \tilde{\boldsymbol{\Phi}}_B \tilde{\boldsymbol{\Theta}}\|_2^2$. From Theorem 4 and 5, we conclude $\{\tilde{\boldsymbol{\Theta}}^k\}$ converges linearly to $\tilde{\boldsymbol{\Theta}}^\star$.

Term (3) is the approximation error caused by RF mapping, which is bounded by

$$
\begin{aligned}
\hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\theta}^\star}) - \hat{\mathcal{E}}(\hat{f}_{\boldsymbol{\alpha}^\star}) &= \sum_{i=1}^{N} \|\tilde{\mathbf{y}}_i - (\tilde{\boldsymbol{\Phi}}_L^i)^\top \boldsymbol{\theta}^\star\|_2^2 - \sum_{i=1}^{N} \|\tilde{\mathbf{y}}_i - \tilde{\mathbf{K}}_i \boldsymbol{\alpha}^\star\|_2^2 \\
&= \|\tilde{\mathbf{y}} - \tilde{\mathbf{f}}_{\boldsymbol{\theta}^\star}\|_2^2 - \|\tilde{\mathbf{y}} - \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star}\|_2^2 \\
&= \|(\tilde{\mathbf{y}} - \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star}) + (\tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star} - \tilde{\mathbf{f}}_{\boldsymbol{\theta}^\star})\|_2^2 - \|\tilde{\mathbf{y}} - \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star}\|_2^2 \\
&= \inf_{\|\tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|} \Big(\|\tilde{\mathbf{y}} - \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star}\|_2^2 + \|\tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star} - \tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|_2^2 + 2\langle \tilde{\mathbf{y}} - \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star}, \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star} - \tilde{\mathbf{f}}_{\boldsymbol{\theta}} \rangle \Big) - \|\tilde{\mathbf{y}} - \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star}\|_2^2 \\
&= \inf_{\|\tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|} \|\tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star} - \tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|_2^2 + 2 \inf_{\|\tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|} \langle \tilde{\mathbf{y}} - \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star}, \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star} - \tilde{\mathbf{f}}_{\boldsymbol{\theta}} \rangle \\
&\leq \inf_{\|\tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|} \|\tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star} - \tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|_2^2 + 2\langle \tilde{\mathbf{y}} - \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star}, \tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star} - \tilde{\mathbf{f}}_{\boldsymbol{\theta}^\star} \rangle \\
&= \inf_{\|\tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|} \|\tilde{\mathbf{f}}_{\boldsymbol{\alpha}^\star} - \tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|_2^2 \\
&\leq \sup_{\|\tilde{\mathbf{f}}_{\mathbf{x}}\|} \inf_{\|\tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|} \|\tilde{\mathbf{f}}_{\mathbf{x}} - \tilde{\mathbf{f}}_{\boldsymbol{\theta}}\|_2^2 \\
&\leq 2\lambda,
\end{aligned}
\tag{2.47}
$$

where the seventh equality comes from Lemma 3 while the last inequality comes from Theorem 8 with $\tilde{\mathbf{f}}_{\mathbf{x}} := [\frac{1}{\sqrt{T_1}}\mathbf{f}_1; \ldots; \frac{1}{\sqrt{T_N}}\mathbf{f}_N] \in \mathbb{R}^T$ and $\mathbf{f}_i = [f(\mathbf{x}_{i,1}), \ldots, f(\mathbf{x}_{i,T_i})]^\top \in \mathbb{R}^{T_i}$ for $f \in \mathcal{H}$.

To bound term (5) of the approximation error of the models in the RKHS $\mathcal{H}$, we refer to the following lemma.

Lemma 4. *[56, Modified Lemma 5] For the kernel κ that can be represented as* (2.10) *and bounded RF mapping, that is $\|\phi(\mathbf{x}, \boldsymbol{\omega})\| \leq 1$ for any $\mathbf{x} \in \mathcal{X}$, under Assumption 3, the*

following holds for any regularization parameter $\lambda > 0$,

$$\mathcal{E}(\hat{f}_{\boldsymbol{\alpha}^\star}) - \mathcal{E}(f_{\mathcal{H}}) = \|\hat{f}_{\boldsymbol{\alpha}^\star} - Pf_p\|_{p_{\mathcal{X}}}^2 \leq (R\lambda^r)^2.$$

In Lemma 4, f_p is the ideal minimizer given the prior knowledge of the marginal distribution $p_{\mathcal{X}}$ of $\mathbf{x}$ and P is a projection operator on f_p so that Pf_p is the optimal minimizer in RKHS. The parameter $r \in [1/2, 1)$ is equivalent to assuming $f_{\mathcal{H}}$ exits, and R can take value as either 1 or $\|\hat{f}_{\boldsymbol{\alpha}^\star}\|_{p_{\mathcal{X}}}$. Setting $r = 1/2$ and $R = 1$, we have

$$\mathcal{E}(\hat{f}_{\boldsymbol{\alpha}^\star}) - \mathcal{E}(f_{\mathcal{H}}) \leq \lambda. \tag{2.48}$$

Combining (2.44)-(2.48) gives

$$\begin{aligned}
\lim_{k\to\infty} \mathcal{E}(\hat{f}^k) - \mathcal{E}(f_{\mathcal{H}})) &\leq \lim_{k\to\infty} \left[\frac{C_1}{\sqrt{T}} + C_3\|\tilde{\boldsymbol{\Theta}}^k - \tilde{\boldsymbol{\Theta}}^\star\|_2 + 2\lambda + \frac{C_2}{\sqrt{T}} + \lambda\right] \\
&= \lim_{k\to\infty} \left[3\lambda + \frac{C_1 + C_2}{\sqrt{T}} + C_3\|\tilde{\boldsymbol{\Theta}}^k - \tilde{\boldsymbol{\Theta}}^\star\|_2\right] \qquad (2.49) \\
&= 3\lambda + O(\frac{1}{\sqrt{T}}),
\end{aligned}$$

and completes the proof of Theorem 3. ■

Chapter 3: Quantized and Communication-Censored Online Decentralized Kernel Learning via Linearized ADMM

3.1 Introduction

In Chapter 2, we developed two algorithms (DKLA and COKE) to tackle the decentralized kernel learning problem under the consensus optimization framework, with COKE further reducing the communication overhead by skipping unnecessary communications. However, both DKLA and COKE operate in batch-form where they assume that all data are available at local agents when the training starts. Whereas in many real-life applications such as wearable devices that collect health statistics from sequential data, or online spam detection [22, 23], data usually come in a streaming manner. In these cases, function learning tasks are expected to perform in an online fashion. Here, the term *online* captures the fact that the data is received by agents in a streaming sequence and agents process the data adaptively.

Online (convex) learning problems have been widely studied in the literature of machine learning, and have been proven to be powerful in making sequential decisions [92, 92]. This scheme can be viewed as a game between a learner (an algorithm) and an adversary (a loss function). Consider only one agent, then at each time instant t, a data pair $\{\mathbf{x}_t, y_t\}(t \in [T])$ arrives to this agent, then learner (algorithm) selects an action $\mathbf{a}_t \in \mathbb{R}^p$, and a loss $\mathcal{L}_t(\mathbf{a}_t)$ parameterized by $\mathbf{a}_t$ is revealed to the learner by the adversary. The main objective of an online learning algorithm is to seek a sequence of actions $\{\mathbf{a}_t, t \in [T]\}$ over the time horizon $t = 1, \ldots, T$, such that the *cumulative regret* is minimized:

$$\mathbf{Reg}_T^S = \sum_{t=1}^{T} \mathcal{L}_t(\mathbf{a}_t) - \sum_{t=1}^{T} \mathcal{L}_t(\mathbf{a}^\star), \tag{3.1}$$

where $\mathbf{a}^\star = \arg\min_{\mathbf{a}\in\mathbb{R}^p} \sum_{t=1}^T \mathcal{L}_t(\mathbf{a}_t)$. Since $\mathbf{a}^\star$ is the fixed single best action in hindsight, we also refer the cumulative regret as *static regret*. Note that actions $\mathbf{a}_t$ are usually assumed to lie in a Euclidean space $\mathbb{R}^p$ in standard online convex optimization problems, which is too optimistic. Motivated by the wide applications of non-parametric kernel methods in various learning tasks [13], in contrast to assume actions $\mathbf{a}_t$ are linear parameters, we hypothesize that the actions are nonlinear functions f_t ($t \in [T]$) that lie in the reproducing kernel Hilbert space $\mathcal{H}$. In this way, the static regret in (3.1) is customized to

$$\mathbf{Reg}_T^S = \sum_{t=1}^T \mathcal{L}_t(f_t) - \sum_{t=1}^T \mathcal{L}_t(f^\star), \tag{3.2}$$

where $f^\star = \arg\min_{f\in\mathcal{H}} \sum_{t=1}^T \mathcal{L}_t(f)$.

According to the Representer theorem [77], the estimate of f_t at time instant t admits

$$\hat{f}_t(\mathbf{x}) = \sum_{\tau=1}^{t-1} \alpha_\tau \kappa(\mathbf{x}, \mathbf{x}_\tau), \tag{3.3}$$

where α_τ's are some coefficients to be optimized given all received data pairs $\{\mathbf{x}_\tau, y_\tau\}_{\tau=1}^{t-1}$.

To learn the nonlinear relationship f_t "on the fly", we then need to estimate α_τ's at each time instant when new data comes. Note that the number of coefficients α_τ needs to be estimated increases across time, making online kernel learning challenging even for a single agent. For many decentralized systems such as mobile sensor networks, the data observed by an agent are kept private without being communicated with others [3,5]. Thus, it is not easy to directly implement traditional kernel methods in a fully decentralized multi-agent setting for online learning. To circumvent these issues, we adopt the random feature (RF) mapping technique to approximate the nonlinear function $\hat{f}_t$ by a fixed size parameter $\boldsymbol{\theta}_t \in \mathbb{R}^{2L}$ in the RF space. Agents in the distributed network thus iteratively communicate and compute the local copy $\boldsymbol{\theta}_{i,t}$ of $\boldsymbol{\theta}_t$ with their neighbors within a one-hop local communication range.

The goal of decentralized online learning turns to minimizing the static regret defined as the difference between the sum of the online loss over all agents and across time suffered by the learning algorithm and that by the best model in hindsight:

$$\mathbf{Reg}_T^S = \sum_{t=1}^{T} \sum_{i=1}^{N} \left(\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}) - \mathcal{L}_{i,t}(\boldsymbol{\theta}^\star) \right), \tag{3.4}$$

where $\boldsymbol{\theta}^\star = \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^{2L}} \sum_{t=1}^{T} \sum_{i=1}^{N} \mathcal{L}_{i,t}(\boldsymbol{\theta})$.

Still, as discussed in Chapter 2, the iterative communications among agents introduce considerable communication overhead. Although the computing capability keeps rising to reduce the computation cost nowadays, a computation-efficient algorithm is still desired to accelerate the learning process for online decision making. In this chapter, we thus focus on decentralized online kernel learning problems in distributed multi-agent systems and aim to develop both communication- and computation-efficient algorithms to minimize the static regret (3.4).

3.1.1 Related work

This work lies at the intersection of (distributed) online convex optimization, scalable online kernel learning, and efficient communication-computation schemes. Related work to these three subjects is reviewed below.

Online convex optimization. Online convex optimization has been widely studied, both in centralized [93–96] and distributed scenarios [97–101]. With an online gradient descent based algorithm [92] or through an online alternating direction method of multipliers (ADMM) [98], a static regret $\mathcal{O}(\sqrt{T})$ can be achieved over a time horizon T. Further, if the cost functions are strictly convex, an efficient algorithm based on the Newton method that achieves a regret bound of $\mathcal{O}(\log T)$ [94]. In addition to static environments, online learning in dynamic environments has attracted more and more attention recently [102–106].

Scalable online kernel learning. To mitigate the curse of dimensionality issue in online

kernel learning, various techniques are developed, including stochastic approximation [37], restricting the number of function parameters [41, 42, 107], and approximating the kernel during training [45, 47, 52]. Among them, RF mapping methods have gained popularity thanks to their ability to map the large-scale data into an RF space of much-reduced dimension by approximating the kernel with a fixed (small) number of random features, which circumvents the curse of dimensionality problem [47, 50, 52]. For the online decentralized kernel learning problem relevant to our work [58,108], gradient descent is conducted locally for each agent to update its learning model, followed by diffusion-based information exchange among agents. Quantization is also considered in [108] to improve communication efficiency. Both works achieve a sublinear regret in the RF space for the online decentralized kernel learning problem. In addition to gradient descent, ADMM is also utilized for online decentralized kernel learning problems [109], where both single kernel learning and multi-kernel learning are studied.

Computation- and communication-efficient optimization. Many methods have been proposed to improve the computation efficiency in decentralized learning, such as [110–113] and references therein. While [111–113] are all based on distributed (sub)gradient descent, they either utilize the gradients of the last two iterates, harness the function smoothness, or leverage the Nesterov gradient, to accelerate the convergence speed. [110] proposes to linearize ADMM to reduce the computation burden of the standard ADMM method while enjoying fast convergence speed. For the iterative learning process, fast convergence also contributes to low communication costs. Other methods to reduce the communication cost include transmitting the compressed information by quantization [60, 62] or sparsification [63, 66]. However, these methods only reduce the required bandwidth at each communication round, not the number of rounds or the number of transmissions. Alternatively, some works suggest randomly selecting a number of nodes for broadcasting/communication and operating asynchronous updating to reduce the number of transmissions per iteration [7, 21, 68, 70–72]. In contrast to random nodes selection, a more intuitive way is to evaluate the importance of a message to avoid unnecessary transmissions [10, 18–20, 59, 73].

This is usually implemented by designing a rule to trigger the communications or adopting a communication censoring scheme to adaptively decide if a message is informative enough to be transmitted during the iterative optimization process.

3.1.2 Contributions

Online kernel learning is challenging even for a single agent because the kernel space is data-centric and evolves as new data arrives. To circumvent this obstacle, we first formulate the online decentralized kernel learning problem as an online consensus optimization problem in the random feature (RF) space. In contrast to solving this problem with standard ADMM as in Chapter 2, we propose to solve it by using linearized ADMM and develop the **O**nline **D**ecentralized **K**ernel learning via **L**inearized **A**DMM (ODKLA) algorithm. In ODKLA, the local cost function of each agent is replaced by its first-order approximation centered at the current iterate, which results in a closed-form primal update if the local cost function is convex. In this way, the computation efficiency of ODKLA is improved compared with the standard ADMM where the primal update involves solving a suboptimization problem every time. To further reduce the communication cost, we develop the **Q**uantized and **C**ommunication-censored **O**nline **D**ecentralized **K**ernel learning via **L**inearized **A**DMM (QC-ODKLA) algorithm by introducing a communication censoring strategy and a quantization strategy. The communication censoring strategy allows each agent to autonomously skip unnecessary communications when its local update is not informative enough for transmission. The quantization strategy restricts the total number of bits transmitted in the learning process. Our key contributions are summarized as follows.

- We develop the ODKLA that utilizes linearized ADMM to solve the online decentralized multi-agent kernel learning problem in the RF space. ODKLA is fully decentralized and does not involve solving sub-optimization problems, thus is more computationally efficient compared with a standard ADMM algorithm.

- We develop the QC-ODKLA algorithm, which utilizes both communication-censoring and quantization strategies and achieves desired learning performance given limited

communication resources and energy supply. When both strategies are absent, QC-ODKLA degenerates to ODKLA.

- We analyze the regret bound of QC-ODKLA and show that QC-ODKLA achieves the optimal sublinear regret $\mathcal{O}(\sqrt{T})$ over T time slots under mild conditions.

- Finally, we test the performance of our proposed ODKLA and QC-ODKLA algorithms on extensive real datasets. The results corroborate that both ODKLA and QC-ODKLA exhibit attractive learning performance and computation efficiency. In addition, QC-ODKLA is highly communication-efficient. Such salient features make it an attractive solution for broad applications where decentralized learning from streaming data is at its core.

Organization. The remaining of this chapter is organized as follows. Section 3.2 formulates the online decentralized kernel learning problem. Section 3.3 develops the online decentralized kernel learning algorithms, including both ODKLA and QC-ODKLA. Section 3.4 presents the theoretical results and Section 3.5 reports the numerical tests using real datasets. Concluding remarks are summarized in Section 3.6.

3.2 Problem Statement

Consider the network and communication model described in Section 1.2. Assume that each agent in the network only has access to its locally observed data composed of independently and identically distributed (i.i.d) input-label pairs $\{\mathbf{x}_{i,t}, y_{i,t}\}_{t=1}^{T}$ obeying an unknown probability distribution p on $\mathcal{X} \times \mathcal{Y}$, with $\mathbf{x}_{i,t} \in \mathbb{R}^d$ and $y_{i,t} \in \mathbb{R}$. The decentralized learning task is to find a nonlinear prediction function f such that $y_{i,t} = f(\mathbf{x}_{i,t}) + e_{i,t}$ for $\{\{\mathbf{x}_{i,t}, y_{i,t}\}_{t=1}^{T}\}_{i=1}^{N}$, where the error term $e_{i,t}$ is minimized accordingly to certain optimality metric. When all data are available offline, and the nonlinear function f belongs to the reproducing kernel Hilbert space (RKHS) $\mathcal{H} := \{f | f(\mathbf{x}) = \sum_{t=1}^{\infty} \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t)\}$ induced by a shift-invariant positive semidefinite kernel $\kappa(\mathbf{x}, \mathbf{x}_t) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, decentralized kernel learning can be

achieved by minimizing the empirical risk (2.2) - (2.3), which we rewrite as follows:

$$f^\star = \arg\min_{f\in\mathcal{H}} \quad \sum_{i=1}^{N}\sum_{t=1}^{T} \ell(f(\mathbf{x}_{i,t}), y_{i,t}) + \lambda\|f\|_{\mathcal{H}}^2, \tag{3.5}$$

where $\ell(\cdot,\cdot)$ is a nonnegative loss function and $\lambda > 0$ is a regularization parameter that controls over-fitting.

As discussed in Chapter 2 and in Section 3.1, directly utilizing the Representer theorem to learn f as a data-centric function is computationally intense. Thus, we utilize the RF mapping method to approximate the function to be learned in (3.5) as

$$f(\mathbf{x}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}_L(\mathbf{x}), \tag{3.6}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{2L}$ is the decision vector to be learned in the RF space, and $\boldsymbol{\phi}_L(\mathbf{x})$ is the mapped data in the RF space given by

$$\boldsymbol{\phi}_L(\mathbf{x}) := \sqrt{\tfrac{1}{L}}[\phi(\mathbf{x},\boldsymbol{\omega}_1), \ldots, \phi(\mathbf{x},\boldsymbol{\omega}_L)]^\top, \tag{3.7}$$

with $\phi(\mathbf{x},\boldsymbol{\omega})$ defined as

$$\phi(\mathbf{x},\boldsymbol{\omega}) = [\cos(\boldsymbol{\omega}^\top\mathbf{x}), \sin(\boldsymbol{\omega}^\top\mathbf{x})]^\top, \tag{3.8}$$

and $\{\boldsymbol{\omega}_l\}_{l=1}^L$ are randomly drawn from $p_\kappa(\boldsymbol{\omega})$, which is the inverse Fourier transform of κ. For a Gaussian kernel $\kappa(\mathbf{x}_t,\mathbf{x}_\tau) = \exp(-\|\mathbf{x}_\mathrm{t} - \mathbf{x}_\tau\|_2^2/(2\sigma^2))$, we have $p_\kappa(\boldsymbol{\omega}) \sim \mathcal{N}(\mathbf{0}, \sigma^{-2}\mathbf{I})$.

With the approximation (3.6), the decentralized kernel learning problem is formulated in Chapter 2 as (2.16) and solved by ADMM-based methods (DKLA and COKE). However, both DKLA and COKE operate in batch-form when all data are available. Whereas in many real-life applications, function learning tasks are expected to perform in an online fashion with sequentially arriving data.

In this chapter, we consider the case that each agent $i(i \in \mathcal{N})$ receives the data points

$\{\mathbf{x}_{i,t}, y_{i,t}\}_{t=1}^{T}$ in an online fashion, and the parameter is estimated based on instantaneous data samples. That is, at each time instant t, each agent i will receive a new data pair $\mathcal{S}_{i,t} := (\mathbf{x}_{i,t}, y_{i,t})$, which will be used to incur a local cost $\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}; \mathcal{S}_{i,t}) := \ell(\boldsymbol{\theta}_{i,t}^{\top}\boldsymbol{\phi}_L(\mathbf{x}_{i,t}), y_{i,t}) + \frac{\lambda}{N}\|\boldsymbol{\theta}_{i,t}\|_2^2$. Note that, this cost depends on the new data only. With the associated local copy $\boldsymbol{\theta}_{i,t}$ of $\boldsymbol{\theta}_t$, the local cost $\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}; \mathcal{S}_{i,t})$, and information from their neighbors, i.e., $\boldsymbol{\theta}_{j,t}, \forall j \in \mathcal{N}_i$, agent i then update the estimate of $\boldsymbol{\theta}_{i,t+1}$.

To achieve an optimal sublinear regret, we formulate decentralized online kernel learning as the following optimization problem:

$$\begin{aligned} \min_{\{\boldsymbol{\theta}_i, \boldsymbol{z}_{ij}\}} \ & \sum_{i=1}^{N} \mathcal{L}_{i,t}(\boldsymbol{\theta}_i; \mathcal{S}_{i,t}) + \frac{\eta_t}{2} \sum_{i=1}^{N} \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i,t}\|_2^2 \\ \text{s.t.} \quad & \boldsymbol{\theta}_i = \boldsymbol{z}_{ij}, \boldsymbol{\theta}_j = \boldsymbol{z}_{ij}, \ \forall (i,j) \in \mathcal{A}, \end{aligned} \tag{3.9}$$

where $\boldsymbol{z}_{ij}$ is an auxiliary variable that enforces consensus on neighboring agents, and the term $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i,t}\|_2^2$ captures the influence from all the past data. At every time t, decentralized online kernel learning (approximately) solves an optimization problem to obtain the update $\boldsymbol{\theta}_{i,t+1}$ from the current decision $\boldsymbol{\theta}_{i,t}$ and the new data $\mathcal{S}_{i,t}$. The formulation (3.9) is customized from the general online decentralized learning problem [98].

In the next section, we first propose a computation-efficient algorithm to solve (3.9). We then utilize communication-censoring and quantization strategies to improve the communication efficiency of the proposed algorithm.

3.3 Algorithm Development

In this section, we first utilize linearized ADMM to efficiently solve (3.9). For notational clarity, we define $\boldsymbol{\Theta} = [\boldsymbol{\theta}_1^{\top}; \boldsymbol{\theta}_2^{\top}; \ldots; \boldsymbol{\theta}_N^{\top}] \in \mathbb{R}^{N \times 2L}$ that contains all the local copies $\boldsymbol{\theta}_i$ and $\boldsymbol{Z} = [\cdots; \boldsymbol{z}_{ij}^{\top}; \cdots] \in \mathbb{R}^{2r \times 2L}$ that contains all auxiliary variables $\boldsymbol{z}_{ij}$. We further define the aggregated function as $\mathcal{L}_t(\boldsymbol{\Theta}; \mathcal{S}_t) := \sum_{i=1}^{N} \mathcal{L}_{i,t}(\boldsymbol{\theta}_i; \mathcal{S}_{i,t})$. With these definitions, we rewrite

(3.9) in a matrix form for the update of $\boldsymbol{\Theta}_{t+1}$ at time t:

$$\begin{aligned} \min_{\{\boldsymbol{\Theta},\boldsymbol{Z}\}} \quad & \mathcal{L}_t(\boldsymbol{\Theta};\mathcal{S}_t) + \frac{\eta_t}{2}\|\boldsymbol{\Theta}-\boldsymbol{\Theta}_t\|_F^2 \\ \text{s.t.} \quad & \boldsymbol{A}\boldsymbol{\Theta}+\boldsymbol{B}\boldsymbol{Z}=\boldsymbol{0}, \end{aligned} \tag{3.10}$$

where $\boldsymbol{A}=\frac{1}{2}[\mathbf{S}_+^\top+\mathbf{S}_-^\top;\mathbf{S}_+^\top-\mathbf{S}_-^\top]\in\mathbb{R}^{4r\times N}$ and $\boldsymbol{B}=[-\boldsymbol{I}_{2r};-\boldsymbol{I}_{2r}]$. Here $\mathbf{S}_+\in\mathbb{R}^{N\times 2r}$ and $\mathbf{S}_-\in\mathbb{R}^{N\times 2r}$ are the unsigned incidence matrix and the signed incidence matrix of the communication graph, respectively.

3.3.1 ODKLA: online decentralized kernel learning via linearized ADMM

The optimization problem (3.10) can be solved via ADMM. To explain, we first get the augmented Lagrangian form of (3.10) as

$$\mathbb{L}_t(\boldsymbol{\Theta},\boldsymbol{Z},\boldsymbol{\Lambda})=\mathcal{L}_t(\boldsymbol{\Theta};\mathcal{S}_t)+\tfrac{\eta_t}{2}\|\boldsymbol{\Theta}-\boldsymbol{\Theta}_t\|_F^2+\langle\boldsymbol{\Lambda},\boldsymbol{A}\boldsymbol{\Theta}+\boldsymbol{B}\boldsymbol{Z}\rangle+\tfrac{\rho}{2}\|\boldsymbol{A}\boldsymbol{\Theta}+\boldsymbol{B}\boldsymbol{Z}\|_F^2, \tag{3.11}$$

where ρ is the penalty parameter or stepsize, $\boldsymbol{\Lambda}=[\boldsymbol{\beta};\boldsymbol{\lambda}]\in\mathbb{R}^{4r\times 2L}$ is the Lagrange multiplier associated with the constraint $\boldsymbol{A}\boldsymbol{\Theta}+\boldsymbol{B}\boldsymbol{Z}=\boldsymbol{0}$. Then, at time t, the updates of the primal variables $\boldsymbol{\Theta},\boldsymbol{Z}$ and the dual variable $\boldsymbol{\Lambda}_t$ are respectively given by

$$\boldsymbol{\Theta}_{t+1}:=\arg\min_{\boldsymbol{\Theta}}\mathbb{L}_t(\boldsymbol{\Theta},\boldsymbol{Z}_t,\boldsymbol{\Lambda}_t), \tag{3.12}$$

$$\boldsymbol{Z}_{t+1}:=\arg\min_{\boldsymbol{Z}}\mathbb{L}_t(\boldsymbol{\Theta}_{t+1},\boldsymbol{Z},\boldsymbol{\Lambda}_t), \tag{3.13}$$

$$\boldsymbol{\Lambda}_{t+1}=\boldsymbol{\Lambda}_t+\rho(\boldsymbol{A}\boldsymbol{\Theta}_{t+1}+\boldsymbol{B}\boldsymbol{Z}_{t+1}). \tag{3.14}$$

Note that given the instantaneous loss $\mathcal{L}_t$, iterates (3.12)-(3.14) only run once, the optimization problem in (3.10) is thus only approximately solved. It has been proven in [98] that with initializations $\boldsymbol{\beta}_1=-\boldsymbol{\lambda}_1$, and $\boldsymbol{Z}_1=\frac{1}{2}\mathbf{S}_+^\top\boldsymbol{\Theta}_1$, the update of the auxiliary variable $\boldsymbol{Z}_t$ is not necessary and the Lagrange multiplier $\boldsymbol{\Lambda}$ can be replaced by a lower dimensional

variable $\mathbf{\Gamma} := [\boldsymbol{\gamma}_1^\top; \cdots; \boldsymbol{\gamma}_N^\top] \in \mathbb{R}^{N \times 2L}$. The explicit updating rules via ADMM are provided in [98].

To further simplify the computation, here we utilize the linearized ADMM to develop the **O**nline **D**ecentralized **K**ernel learning via **L**inearized **A**DMM (ODKLA) algorithm. Specifically, we replace $\mathcal{L}_t(\mathbf{\Theta})$ in (3.12) by its linear approximation $\mathcal{L}_t(\mathbf{\Theta}_t) + \langle \partial \mathcal{L}_t(\mathbf{\Theta}_t), \mathbf{\Theta} - \mathbf{\Theta}_t \rangle$ at $\mathbf{\Theta} = \mathbf{\Theta}_t$. In this way, the iterates of $\mathbf{\Theta}_{t+1}$ and $\mathbf{\Gamma}_{t+1}$ can be generated by the simplified recursions:

$$\mathbf{\Theta}_{t+1} = (\eta_t \mathbf{I} + 2\rho \mathbf{D})^{-1} \Big[(\rho(\mathbf{D} + \mathbf{W}) + \eta_t \mathbf{I}) \mathbf{\Theta}_t - \mathbf{\Gamma}_t - \partial \mathcal{L}_t(\mathbf{\Theta}_t) \Big], \tag{3.15}$$

$$\mathbf{\Gamma}_{t+1} = \mathbf{\Gamma}_t + \rho(\mathbf{D} - \mathbf{W}) \mathbf{\Theta}_{t+1}. \tag{3.16}$$

For distributed implementation of the ODKLA algorithm, each agent i only needs to update a primal variable $\boldsymbol{\theta}_i$ and a dual variable $\boldsymbol{\gamma}_i$ with the following iterations:

$$\boldsymbol{\theta}_{i,t+1} = \boldsymbol{\theta}_{i,t} - \frac{1}{\eta_t + 2\rho d_i} \Big[\partial \mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}) + \rho \sum_{j \in \mathcal{N}_i} (\boldsymbol{\theta}_{i,t} - \boldsymbol{\theta}_{j,t}) + \boldsymbol{\gamma}_{i,t} \Big], \tag{3.17}$$

$$\boldsymbol{\gamma}_{i,t+1} = \boldsymbol{\gamma}_{i,t} + \rho \sum_{j \in \mathcal{N}_i} (\boldsymbol{\theta}_{i,t+1} - \boldsymbol{\theta}_{j,t+1}). \tag{3.18}$$

Note that with linearized ADMM, at each time t, ODKLA has closed-form solutions for all agents to update their primal variables, instead of solving optimization problems as in (3.12). Thus, the computational efficiency is improved. The ODKLA algorithm is outlined in Algorithm 3.

3.3.2 QC-ODKLA: quantized and communication-censored ODKLA

ODKLA resolves the challenges caused by streaming data in decentralized kernel learning in a computationally efficient manner. However, as seen in (3.17) - (3.18), agents communicate all the time, resulting in low communication efficiency. Thus, we introduce

Algorithm 3 ODKLA (run at agent i)

Require: Kernel κ, hyper-parameters (L, ρ, η_t), initialize local variables to $\boldsymbol{\theta}_{i,1} = \mathbf{0}$, and $\boldsymbol{\gamma}_{i,1} = 0$.

1: Draw L i.i.d. samples $\{\boldsymbol{\omega}_l\}_{l=1}^L$ from $p_\kappa(\boldsymbol{\omega})$ according to a common random seed.
2: **for** iterations $t = 1, 2, \ldots, T$ **do**
3: Receive a streaming data $(\mathbf{x}_{i,t}, y_{i,t})$.
4: Construct $\boldsymbol{\phi}(\mathbf{x}_{i,t})$ via (3.7).
5: Update local primal variable $\boldsymbol{\theta}_{i,t+1}$ via (3.17).
6: Transmit $\boldsymbol{\theta}_{i,t+1}$ to neighbors and receive $\boldsymbol{\theta}_{j,t+1}$ from neighbors $j \in \mathcal{N}_i$.
7: Update local dual variable $\boldsymbol{\gamma}_{i,t+1}$ via (3.18).
8: **end for**

communication censoring and quantization strategies to cope with the limited communication resource situation. The resulting algorithm is **Q**uantized and **C**ommunication-censored **O**nline **D**ecentralized **K**ernel learning via **L**inearized **A**DMM (QC-ODKLA).

As in Chapter 2, we first introduce a new state variable $\hat{\boldsymbol{\theta}}_{i,t}$ for each agent i to record its latest broadcast primal variable up to time t. Then, the difference between agent i's updated state $\boldsymbol{\theta}_{i,t+1}$ and its previously transmitted state $\hat{\boldsymbol{\theta}}_{i,t}$ at time t is defined as

$$\boldsymbol{h}_{i,t} = \boldsymbol{\theta}_{i,t+1} - \hat{\boldsymbol{\theta}}_{i,t}, \tag{3.19}$$

and the evaluation function that evaluates if the local updates $\boldsymbol{\theta}_{i,t+1}$ are informative enough to be transmitted is given by

$$H_{i,t} = \|\boldsymbol{h}_{i,t}\|_2 - \varrho\mu^t, \tag{3.20}$$

with predefined positive constants $\varrho > 0$ and $\mu < 1$.

We also introduce a quantization scheme to reduce the communication cost from the perspective of bit numbers per transmission, and adopt the difference-based quantization scheme to facilitate the measurement and analysis of the impact of quantization [114]. To be specific, for each element $h_{i,t}^l (l = 1, \ldots, 2L)$ of $\boldsymbol{h}_{i,t}$ within the range of $[u, v)$, if we restrict the number of transmission bits to be b, then we can evenly divided the range $[u, v)$ to be $q = 2^b$ intervals of equal length $\Delta = (v - u)/q$. Then the rounding quantizer $Q(\cdot)$ applied

to $h_{i,t}^l$ outputs

$$Q(h_{i,t}^l) = u + \left(\left\lfloor \frac{h_{i,t}^l - u}{\Delta} \right\rfloor + \frac{1}{2} \right) \Delta, \tag{3.21}$$

where $\lfloor \cdot \rfloor$ is the floor operation. In practice, it is not necessary to transmit $Q(h_{i,t}^l)$, instead, we can simply transmit the integer $k := \left\lfloor \frac{h_{i,t}^l - u}{\Delta} \right\rfloor$ using the b bits. Thus, the total number of bits for agent i to transmit the quantized difference $Q(\boldsymbol{h}_{i,t})$ to its neighbors is only $2Lb$ bits.

The whole communication process thus involves three parts: evaluation, quantization, and state update. If $H_{i,t} \geq 0$, then $\boldsymbol{\theta}_{i,t+1}$ is deemed informative, and agent i is allowed to transmit a quantized difference $Q(\boldsymbol{h}_{i,t})$ to its neighbors and updates its local state as $\hat{\boldsymbol{\theta}}_{i,t+1} = \hat{\boldsymbol{\theta}}_{i,t} + Q(\boldsymbol{h}_{i,t})$. Otherwise, $\boldsymbol{\theta}_{i,t+1}$ is censored, agent i sets $\hat{\boldsymbol{\theta}}_{i,t+1} = \hat{\boldsymbol{\theta}}_{i,t}$, and no information is transmitted. Similarly, upon receiving $Q(\boldsymbol{h}_{j,t})$ from its neighbor j, agent i updates the state variable of its neighbor's as $\hat{\boldsymbol{\theta}}_{j,t+1} = \hat{\boldsymbol{\theta}}_{j,t} + Q(\boldsymbol{h}_{j,t})$, otherwise, sets $\hat{\boldsymbol{\theta}}_{j,t+1} = \hat{\boldsymbol{\theta}}_{j,t}$.

With the communication censoring rule and the quantization scheme, the primal and dual updates in (3.17) and (3.18) become

$$\boldsymbol{\theta}_{i,t+1} = \boldsymbol{\theta}_{i,t} - \frac{1}{\eta_t + 2\rho d_i} \Big[\partial \mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}) + \rho \sum_{j \in \mathcal{N}_i} (\hat{\boldsymbol{\theta}}_{i,t} - \hat{\boldsymbol{\theta}}_{j,t}) + \boldsymbol{\gamma}_{i,t} \Big], \tag{3.22}$$

$$\boldsymbol{\gamma}_{i,t+1} = \boldsymbol{\gamma}_{i,t} + \rho \sum_{j \in \mathcal{N}_i} \Big(\hat{\boldsymbol{\theta}}_{i,t+1} - \hat{\boldsymbol{\theta}}_{j,t+1} \Big). \tag{3.23}$$

Compared with ODKLA, the total numbers of transmissions and bits required by QC-ODKLA are both reduced in the optimization and learning process. We summarize the QC-ODKLA algorithm in Algorithm 4.

Algorithm 4 QC-ODKLA (run at agent i)

Require: Kernel κ, hyper-parameters $(L, \rho, \eta_t, \varrho, \mu)$, initialize local variables to $\boldsymbol{\theta}_{i,1} = \mathbf{0}$, and $\boldsymbol{\gamma}_{i,1} = \mathbf{0}$, $\hat{\boldsymbol{\theta}}_{i,1} = Q(\boldsymbol{\theta}_{i,1})$, and $\hat{\boldsymbol{\theta}}_{j,1} = Q(\boldsymbol{\theta}_{j,1})$ for all $j \in \mathcal{N}_i$.

1: Draw L i.i.d. samples $\{\boldsymbol{\omega}_l\}_{l=1}^L$ from $p_\kappa(\boldsymbol{\omega})$ according to a common random seed.
2: **for** iterations $t = 1, 2, \ldots, T$ **do**
3: Receive a streaming data $(\mathbf{x}_{i,t}, y_{i,t})$.
4: Construct $\boldsymbol{\phi}(\mathbf{x}_{i,t})$ via (3.7).
5: Update local primal variable $\boldsymbol{\theta}_{i,t+1}$ by solving (3.22).
6: Calculate the difference $\boldsymbol{h}_{i,t}$ via (3.19) and quantize it as $Q(\boldsymbol{h}_{i,t})$ via (3.21).
7: If (3.20) is nonnegative, transmit $Q(\boldsymbol{h}_{i,t})$ to neighbors and set $\hat{\boldsymbol{\theta}}_{i,t+1} = \hat{\boldsymbol{\theta}}_{i,t} + Q(\boldsymbol{h}_{i,t})$. Else, set $\hat{\boldsymbol{\theta}}_{i,t+1} = \hat{\boldsymbol{\theta}}_{i,t}$ and do not transmit.
8: If receiving $Q(\boldsymbol{h}_{j,t})$ from neighbors j, update $\hat{\boldsymbol{\theta}}_{j,t+1} = \hat{\boldsymbol{\theta}}_{j,t} + Q(\boldsymbol{h}_{j,t})$. Else, set $\hat{\boldsymbol{\theta}}_{j,t+1} = \hat{\boldsymbol{\theta}}_{j,t}$.
9: Update local dual variable $\boldsymbol{\gamma}_{i,t+1}$ via (3.23).
10: **end for**

3.4 Regret Analysis

In this section, we analyze the regret bound of QC-ODKLA, as defined in (3.4). We prove that QC-ODKLA achieves the optimal sublinear regret $\mathcal{O}(\sqrt{T})$ for convex local cost functions $\mathcal{L}_{i,t}$. Since ODKLA is a special case of QC-ODKLA where both the quantization and communication-censoring strategies are absent, the regret analysis of QC-ODKLA extends to ODKLA straightforwardly. The following commonly used assumptions are adopted.

Assumption 4. *The local cost functions $\mathcal{L}_{i,t}(\boldsymbol{\theta})$ are convex and differentiable with respect to $\boldsymbol{\theta}$. Also, assume the gradient norm is upper bounded by a positive constant G, or equivalently, $\|\partial\mathcal{L}_{i,t}(\boldsymbol{\theta})\|_2 \leq G, \forall i$.*

Assumption 5. *The optimal solution of* (3.9) *is upper bounded by C_θ, i.e., $\|\boldsymbol{\theta}^\star\|_2 \leq C_\theta$.*

Note that all assumptions are standard in online decentralized kernel learning [58, 108, 109]. The convexity of local cost functions are easily satisfied in learning problems if the local cost functions are square loss or the hinge loss.

To study the regret bound for QC-ODKLA, we notice that the difference of QC-ODKLA and ODKLA is the communication censoring step and the quantization step in the communication stage, which introduces an error if an update is censored and/or quantized in a

transmission. We define the introduced error of agent i at time t as

$$\boldsymbol{e}_{i,t} := \boldsymbol{\theta}_{i,t} - \hat{\boldsymbol{\theta}}_{i,t}, \tag{3.24}$$

and the overall introduced error at time t as $\boldsymbol{E}_t := [\boldsymbol{e}_{1,t}^\top; \boldsymbol{e}_{2,t}^\top; \ldots; \boldsymbol{e}_{N,t}^\top]$. We first show that the overall introduced error in QC-ODKLA is upper bounded by the quantization error and the pre-defined threshold parameters.

Lemma 5. *For the updates* (3.22) *and* (3.23)*, under Assumptions 4 and 5, if the quantized difference $Q(\boldsymbol{h}_{i,t})$ is only allowed to transmit when $H_{i,t} \geq 0$ for the pre-defined threshold parameters v and μ, then, for any time $t > 0$, the overall error introduced in the QC-ODKLA is upper bounded by*

$$\|\boldsymbol{E}_t\|_F^2 \leq \zeta := \max\{\sqrt{N}\varrho\mu, \sqrt{2NL}\Delta/2\}, \tag{3.25}$$

where Δ is the length of the quantization interval.

Proof. Define $\delta\hat{\boldsymbol{\theta}}_{i,t} = \hat{\boldsymbol{\theta}}_{i,t} - \hat{\boldsymbol{\theta}}_{i,t-1}$, the introduced error for each agent i is then represented as

$$\begin{aligned} \boldsymbol{e}_{i,t} &= \boldsymbol{\theta}_{i,t} - \hat{\boldsymbol{\theta}}_{i,t} \\ &= \boldsymbol{\theta}_{i,t} - \hat{\boldsymbol{\theta}}_{i,t-1} - \delta\hat{\boldsymbol{\theta}}_{i,t} \\ &= \boldsymbol{h}_{i,t-1} - \delta\hat{\boldsymbol{\theta}}_{i,t}. \end{aligned} \tag{3.26}$$

According to the censoring rule, if $\|\boldsymbol{h}_{i,t-1}\|_2 \geq \varrho\mu^{t-1}$ for $t \geq 1$, we have $\delta\hat{\boldsymbol{\theta}}_{i,t} = Q(\boldsymbol{h}_{i,t-1})$, which implies $\|\boldsymbol{e}_{i,t}\|_2 = \|\boldsymbol{h}_{i,t-1} - Q(\boldsymbol{h}_{i,t-1})\|_2 \leq \sqrt{2L}\Delta/2$. Otherwise, if $\|\boldsymbol{h}_{i,t-1}\|_2 < \varrho\mu^{t-1}$ for $t \geq 1$, we have $\delta\hat{\boldsymbol{\theta}}_{i,t} = \boldsymbol{0}$, which implies $\|\boldsymbol{e}_{i,t}\|_2 = \|\boldsymbol{h}_{i,t-1}\|_2 \leq \varrho\mu^{t-1} \leq \varrho\mu$ since $\mu < 1$. Therefore, the overall introduced error $\|\boldsymbol{E}_t\|_F^2 \leq \max\{\sqrt{N}\varrho\mu, \sqrt{2NL}\Delta/2\}$. ■

With Lemma 5, we are ready to establish the network regret bound of QC-ODKLA.

Theorem 9. *Under the assumptions 4 - 5, if the quantized difference* $Q(\boldsymbol{h}_{i,t})$ *is only allowed to transmit when* $H_{i,t} \geq 0$ *for the pre-defined threshold parameters* $\varrho > 0$ *and* $\mu < 1$*, then, for any time* $t > 0$*, the cumulative network regret* (3.4) *generated by the updates* (3.22) *and* (3.23) *satisfies*

$$\mathbf{Reg}_T^S \leq (\sqrt{N}C_\theta + \frac{\sqrt{N}G}{\sigma_{\max}^2(\mathbf{S}_-)} + \sigma_{\max}^2(\mathbf{S}_-)\zeta)\mathcal{O}(\sqrt{T}) \tag{3.27}$$

if $\eta_t = \rho = 1/\mathcal{O}(\sqrt{T})$.

Proof. See Appendix 3.7.1. ■

Remark 1. Comparing the regret bound generated by QC-ODKLA and ODKLA, we notice that the censoring and quantization strategies also affect the cumulative network regret, in addition to the network size and topology.

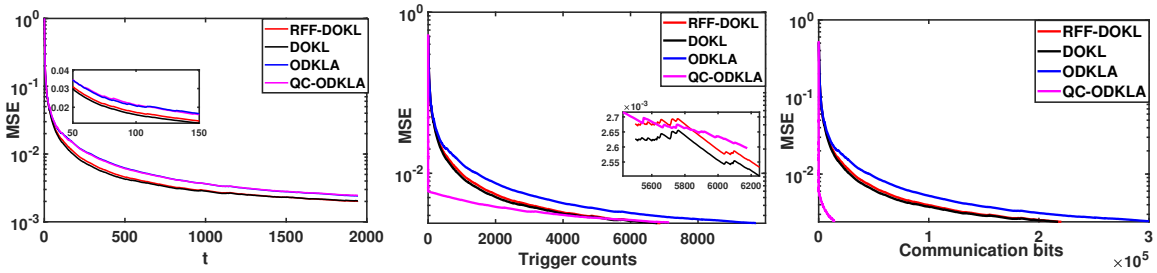
3.5 Experiments

This section evaluates the performance of our proposed ODKLA and QC-ODKLA algorithms in regression tasks for streaming data from real-world datasets.

Benchmarks. Since we consider the case that data are only locally available and cannot be shared among agents, the RFF-DOKL algorithm which is developed based on online gradient descent and a diffusion strategy [58] and the DOKL algorithm which is developed based on online ADMM [109] will be simulated and compared in our experiments with the proposed ODKLA and QC-ODKLA algorithms.

Datasets. The regression tasks are carried out on six datasets available at the UCI machine learning repository [86]: (i) Tom's hardware ($T_{total} = 11000$, $d = 96$); (ii) Twitter ($T_{total} = 98700$, $d = 77$); (iii) Energy ($T_{total} = 18600$, $d = 28$); (iv) Air quality ($T_{total} = 7320$, $d = 28$); (v) Conductivity ($T_{total} = 21260$, $d = 81$); and (vi) Blood ($T_{total} = 61000$, $d = 2$). The detailed description on all datasets is deferred to Appendix 3.7.2.

Settings and parameter tuning. All experiments are conducted using Matlab 2021 on an Intel CPU @ 3.6 GHz (32 GB RAM) desktop. For each dataset, the T_{total} data

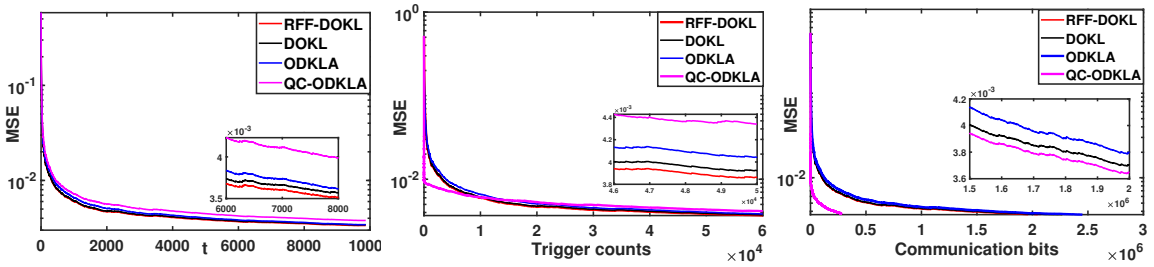


(a) MSE versus time. (b) MSE versus triggers. (c) MSE versus bits.

Figure 3.1: Learning performance on Tom's hardware dataset.

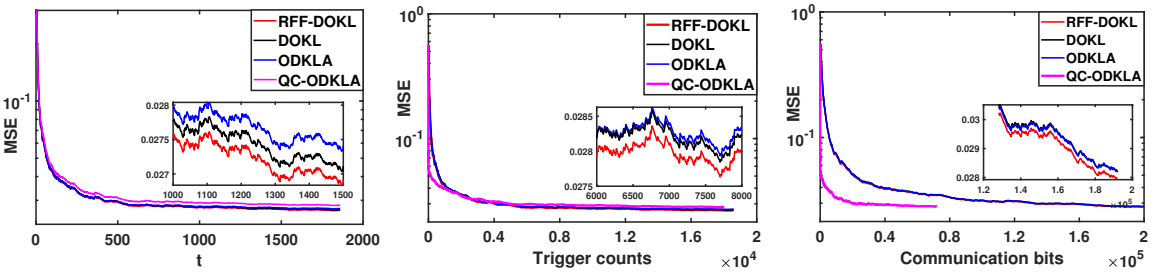
samples are randomly shuffled and then partitioned among N nodes so that each node has $T = T_{total}/N$ samples. The features are normalized so that all values are between 0 and 1. The number of random features adopted for RF approximation is $L = 50$ throughout the simulations. The Gaussian kernel bandwidth is fined tuned to be $\sigma = 0.5$ for Tom's hardware, Twitter, Air quality, and Blood datasets. For Conductivity and Energy datasets, $\sigma = 1$ and 0.1, respectively. The regularization parameter $\lambda = 10^{-4}$. The stepsize ρ and η_t are fine-tuned via grid-search for each method and each dataset individually. The connected graph is randomly generated with $N = 5$ or $N = 10$ nodes. For Twitter, Conductivity, and Blood datasets, we use a 10-node network. The remaining datasets use a 5-node network. The censoring threshold parameters are $\varrho = 2, \mu = 0.9$ for energy data, and $\varrho = 4, \mu = 0.99$ for all the other datasets.

MSE performance. We first evaluate the learning performance of all algorithms by the mean-squared error (MSE), which is commonly adopted in online learning problems [58,109]. From Figures 3.1 (a) - 3.6 (a), we can see that the learning performance of ODKLA, RFF-DOKL, and DOKL is very close while the trivial difference comes from the distinction of specific datasets. Further, the learning performance of QC-ODKLA is always comparable to that of the ODKLA, after introducing the communication censoring and quantization strategies. The quantization level is set to be $q = 8$,



(a) MSE versus time. (b) MSE versus triggers. (c) MSE versus bits.

Figure 3.2: Learning performance on Twitter dataset.

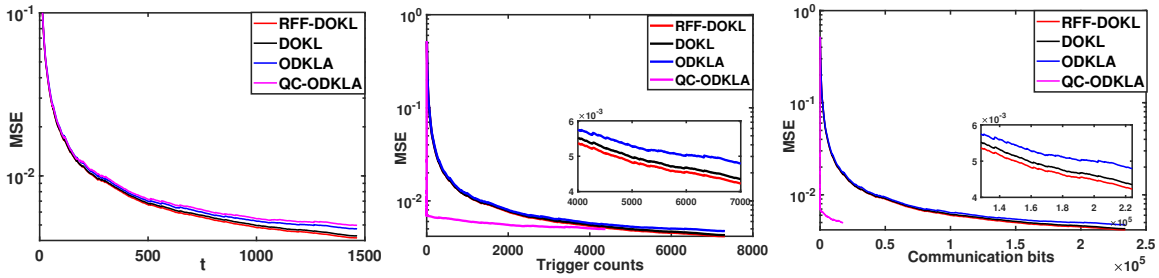


(a) MSE versus time. (b) MSE versus triggers. (c) MSE versus bits.

Figure 3.3: Learning performance on Energy dataset.

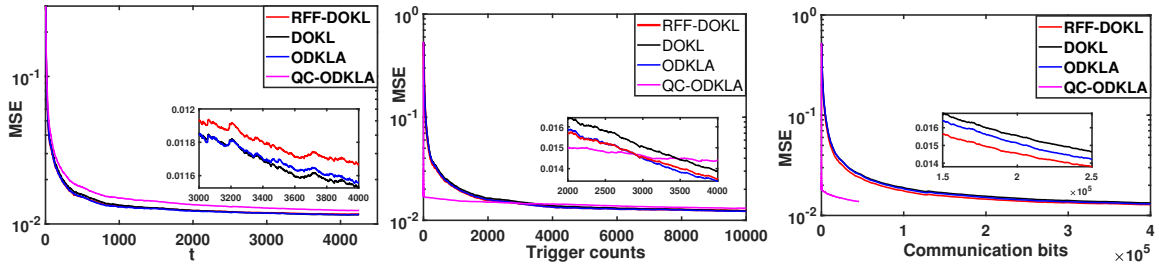
Communication efficiency. We then evaluate the communication efficiency among different algorithms. We present the MSE performance versus trigger counts in Figures 3.1 (b) - 3.6 (b) and MSE performance versus communication bits in Figures 3.1 (c) - 3.6 (c). Figures 3.1 (b) - 3.6 (b) show that QC-ODKLA triggers a few transmissions in the early learning stage, which greatly improves the communication efficiency. Further, thanks to the quantization, QC-ODKLA only needs 3 bits to transmit an element, the total number of communication bits is also greatly reduced accordingly. For other methods to transmit each element of updates, suppose the agent uses a 32-bit CPU operating mode, and then the communication cost is 32 bits per iteration per agent per element. Therefore, QC-ODKLA is corroborated to greatly reduce the communication cost.

Computation efficiency. Finally, we evaluate the computation efficiency of all algorithms



(a) MSE versus time. (b) MSE versus triggers. (c) MSE versus bits.

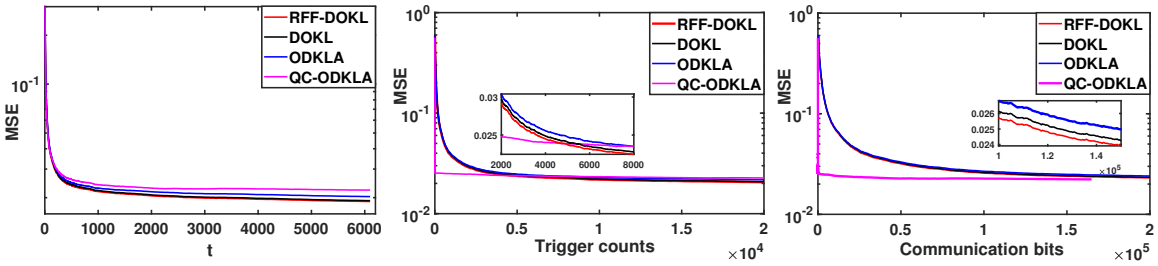
Figure 3.4: Learning performance on Air dataset.



(a) MSE versus time. (b) MSE versus triggers. (c) MSE versus bits.

Figure 3.5: Learning performance on Conductivity dataset.

by their running time on six datasets, which is recorded in Table 3.1. RFF-DOKL is a gradient descent-based first-order algorithm, which achieves the highest computation efficiency. Comparing ODKLA with DOKL, we see that the linearization step reduces a large amount of computation to standard ADMM. Under the circumstance that online streaming data vary fast, a computation-efficient algorithm is preferred, reflecting the advantages of the proposed ODKLA and QC-ODKLA algorithms. Also, note that QC-ODKLA is computationally slower than ODKLA since the communication censoring and quantization steps consume computation resources.



(a) MSE versus time. (b) MSE versus triggers. (c) MSE versus bits.

Figure 3.6: Learning performance on Blood dataset.

Data set	RFF-DOKL	DOKL	ODKLA	QC-ODKLA
Tom's hardware	0.3541s	2.0613s	0.4455s	0.7247s
Twitter	4.6148s	21.8808s	6.2351s	9.4218s
Energy	0.8584s	4.2681s	1.1600s	1.7928s
Air quality	0.2760s	1.6808s	0.4717s	0.5510s
Conductivity	0.7952s	4.4285s	0.9784s	1.6201s
Blood	2.6194s	13.6249s	3.6104s	5.4290s

Table 3.1: The running time of four algorithms on six datasets.

3.6 Concluding Remarks

This chapter studies the online decentralized kernel learning problem under communication constraints for multi-agent systems. We utilize RF mapping to circumvent the curse of dimensionality issue caused by the increasing size of sequentially arriving data. To efficiently solve such a challenging problem, we then develop a novel online decentralized kernel learning algorithm via linearized ADMM (ODKLA). We integrate the communication-censoring and quantization strategies into the proposed ODKAL as QC-ODKLA to further save communication overheads. We derive the sublinear regret bounds for both algorithms theoretically and verify their effectiveness in learning performance and efficiency in both communication and computation via simulations on various real datasets.

3.7 Appendix

3.7.1 Proof of Theorem 9

Proof. Define $\boldsymbol{\Theta}^\star = [\boldsymbol{\theta}^{\star\top}; \ldots; \boldsymbol{\theta}^{\star\top}] \in \mathbb{R}^{N\times 2L}$, which is the stack of N copies of $\boldsymbol{\theta}^\star$, and $\mathcal{L}_t(\boldsymbol{\Theta}^\star) := \sum_{i=1}^N \mathcal{L}_{i,t}(\boldsymbol{\theta}^\star)$, we rewrite (3.4) as

$$\begin{aligned}\mathbf{Reg}_T^S &= \sum_{t=1}^T \left(\sum_{i=1}^N \mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}) - \sum_{i=1}^N \mathcal{L}_{i,t}(\boldsymbol{\theta}^\star) \right) \\ &= \sum_{t=1}^T \left(\mathcal{L}_t(\boldsymbol{\Theta}_t) - \mathcal{L}_t(\boldsymbol{\Theta}^\star) \right).\end{aligned} \tag{3.28}$$

To analyze the regret bound of QC-ODKLA, we first represent the matrix form of QC-ODKLA updates (3.22) - (3.23) as

$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t - (\eta_t \mathbf{I} + 2\rho\mathbf{D})^{-1}\Big[\partial\mathcal{L}_t(\boldsymbol{\Theta}_t) + \rho(\mathbf{D}-\mathbf{W})\hat{\boldsymbol{\Theta}}_t + \boldsymbol{\Gamma}_t\Big], \tag{3.29}$$

$$\boldsymbol{\Gamma}_{t+1} = \boldsymbol{\Gamma}_t + \rho(\mathbf{D}-\mathbf{W})\hat{\boldsymbol{\Theta}}_{t+1}, \tag{3.30}$$

where $\hat{\boldsymbol{\Theta}}_t = [\hat{\boldsymbol{\theta}}_{1,t}^\top; \ldots; \hat{\boldsymbol{\theta}}_{N,t}^\top] \in \mathbb{R}^{N\times 2L}$. Note that the censoring and quantization are implemented after step (3.29) and before step (3.30).

For the network model we adopt, according to [115], we have

$$\begin{aligned}\mathbf{D} + \mathbf{W} &= \frac{1}{2}\mathbf{S}_+\mathbf{S}_+^\top, \\ \mathbf{D} - \mathbf{W} &= \frac{1}{2}\mathbf{S}_-\mathbf{S}_-^\top.\end{aligned} \tag{3.31}$$

Using the second equality of (3.31) and denoting the introduced error of all agents $\boldsymbol{E}_t$

as $\boldsymbol{E}_t := \boldsymbol{\Theta}_t - \hat{\boldsymbol{\Theta}}_t$, we obtain the equivalent form of (3.29) and (3.30) respectively as

$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t - (\eta_t \mathbf{I} + 2\rho \mathbf{D})^{-1} \Big[\partial \mathcal{L}_t(\boldsymbol{\Theta}_t) + \frac{\rho}{2} \mathbf{S}_- \mathbf{S}_-^\top \boldsymbol{\Theta}_t - \frac{\rho}{2} \mathbf{S}_- \mathbf{S}_-^\top \boldsymbol{E}_t + \boldsymbol{\Gamma}_t \Big], \tag{3.32}$$

$$\boldsymbol{\Gamma}_{t+1} = \boldsymbol{\Gamma}_t + \frac{\rho}{2} \mathbf{S}_- \mathbf{S}_-^\top \boldsymbol{\Theta}_{t+1} - \frac{\rho}{2} \mathbf{S}_- \mathbf{S}_-^\top \boldsymbol{E}_{t+1}. \tag{3.33}$$

Observe from (3.33) that $\boldsymbol{\Gamma}_{t+1}$ stays in the column space of $\mathbf{S}_- \mathbf{S}_-^\top$ if $\boldsymbol{\Gamma}_1$ is also initialized therein. Therefore, we introduce variables $\boldsymbol{\beta}_t \in \mathbb{R}^{2r \times 2L}$, which stay in the column space of $\mathbf{S}_-^\top$, and let $\boldsymbol{\Gamma}_t = \mathbf{S}_- \boldsymbol{\beta}_t$ for any $t \geq 1$. Then, (3.33) is equivalent to

$$\boldsymbol{\beta}_{t+1} = \boldsymbol{\beta}_t + \frac{\rho}{2} \mathbf{S}_-^\top \boldsymbol{\Theta}_{t+1} - \frac{\rho}{2} \mathbf{S}_-^\top \boldsymbol{E}_{t+1}. \tag{3.34}$$

Using (3.33) and $\boldsymbol{\Gamma}_t = \mathbf{S}_- \boldsymbol{\beta}_t$ to eliminate $\boldsymbol{\Gamma}_t$, we rewrite (3.32) as

$$\boldsymbol{\Theta}_{t+1} = \boldsymbol{\Theta}_t - (\eta_t \mathbf{I} + 2\rho \mathbf{D})^{-1} \Big[\partial \mathcal{L}_t(\boldsymbol{\Theta}_t) + \frac{\rho}{2} \mathbf{S}_- \mathbf{S}_-^\top (\boldsymbol{E}_{t+1} - \boldsymbol{E}_t) + \frac{\rho}{2} \mathbf{S}_- \mathbf{S}_-^\top (\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}) + \mathbf{S}_- \boldsymbol{\beta}_{t+1} \Big]. \tag{3.35}$$

The following analysis is based on the equivalent form of the QC-ODKLA algorithm given by (3.35) and (3.34). The Karush–Kuhn–Tucker (KKT) conditions of (3.10) are

$$\partial \mathcal{L}_t(\boldsymbol{\Theta}^\star) + \eta_t (\boldsymbol{\Theta}^\star - \boldsymbol{\Theta}_t) + \mathbf{S}_- \boldsymbol{\beta}^\star = \mathbf{0}, \tag{3.36a}$$

$$\mathbf{S}_-^\top \boldsymbol{\Theta}^\star = \mathbf{0}, \tag{3.36b}$$

$$\frac{1}{2} \mathbf{S}_+^\top \boldsymbol{\Theta}^\star = \boldsymbol{Z}^\star, \tag{3.36c}$$

where $(\boldsymbol{\Theta}^\star, \boldsymbol{Z}^\star, \boldsymbol{\beta}^\star)$ is the optimal primal-dual triplet.

Rearrange terms in (3.35) to place $\partial\mathcal{L}_t(\boldsymbol{\Theta}_t)$ at the left side, we have

$$\begin{aligned}\partial\mathcal{L}_t(\boldsymbol{\Theta}_t) &= (\eta_t\mathbf{I} + 2\rho\mathbf{D} - \frac{\rho}{2}\mathbf{S}_-\mathbf{S}_-^\top)(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}) + \frac{\rho}{2}\mathbf{S}_-\mathbf{S}_-^\top(\boldsymbol{E}_t - \boldsymbol{E}_{t+1}) - \mathbf{S}_-\boldsymbol{\beta}_{t+1} \\ &= (\eta_t\mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}) + \frac{\rho}{2}\mathbf{S}_-\mathbf{S}_-^\top(\boldsymbol{E}_t - \boldsymbol{E}_{t+1}) - \mathbf{S}_-\boldsymbol{\beta}_{t+1},\end{aligned} \tag{3.37}$$

where the second equality utilizes (3.31) such that $2\mathbf{D} = \frac{1}{2}\mathbf{S}_-\mathbf{S}_-^\top + \frac{1}{2}\mathbf{S}_+\mathbf{S}_+^\top$. We consider to bound the instantaneous regret $\mathcal{L}_t(\boldsymbol{\Theta}_t) - \mathcal{L}_t(\boldsymbol{\Theta}^\star)$ at time t first. With Assumption 4, it holds

$$\mathcal{L}_t(\boldsymbol{\Theta}_t) - \mathcal{L}_t(\boldsymbol{\Theta}^\star) \leq \langle \partial\mathcal{L}_t(\boldsymbol{\Theta}_t), \boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star \rangle. \tag{3.38}$$

Substitute the expression of $\partial\mathcal{L}_t(\boldsymbol{\Theta}_t)$ in (3.37) into (3.38) yields

$$\begin{aligned}\mathcal{L}_t(\boldsymbol{\Theta}_t) - \mathcal{L}_t(\boldsymbol{\Theta}^\star) \leq\ & \langle (\eta_t\mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}), \boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star \rangle \\ & + \langle \frac{\rho}{2}\mathbf{S}_-\mathbf{S}_-^\top(\boldsymbol{E}_t - \boldsymbol{E}_{t+1}) - \mathbf{S}_-\boldsymbol{\beta}_{t+1}, \boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star \rangle.\end{aligned} \tag{3.39}$$

Now we reorganize the two terms on the right-hand side of (3.39). For the first term, we have

$$\begin{aligned}&\langle (\eta_t\mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}), \boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star \rangle \\ &\leq \sigma_{\max}(\eta_t\mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)\langle \boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}, \boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star \rangle \\ &= \frac{\sigma_{\max}(\eta_t\mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2}\Big(\|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star\|_F^2 - \|\boldsymbol{\Theta}_{t+1} - \boldsymbol{\Theta}^\star\|_F^2 + \|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}\|_F^2 \Big),\end{aligned} \tag{3.40}$$

where $\sigma_{\max}(\eta_t\mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)$ denotes the maximum singular value of $\eta_t\mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top$.

For the second term, we have

$$
\begin{aligned}
&\langle \frac{\rho}{2}\mathbf{S}_-\mathbf{S}_-^\top(\boldsymbol{E}_t - \boldsymbol{E}_{t+1}) - \mathbf{S}_-\boldsymbol{\beta}_{t+1}, \boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star\rangle \\
&= \langle \frac{\rho}{2}\mathbf{S}_-^\top(\boldsymbol{E}_t - \boldsymbol{E}_{t+1}) - \boldsymbol{\beta}_{t+1}, \mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star)\rangle \\
&\overset{(a)}{=} \langle \frac{\rho}{2}\mathbf{S}_-^\top(\boldsymbol{E}_t - \boldsymbol{E}_{t+1}) - \boldsymbol{\beta}_{t+1}, \mathbf{S}_-^\top\boldsymbol{\Theta}_t\rangle \\
&\overset{(b)}{=} \langle \frac{\rho}{2}\mathbf{S}_-^\top(\boldsymbol{E}_t - \boldsymbol{E}_{t+1}) - \boldsymbol{\beta}_{t+1}, \frac{2}{\rho}(\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}) + \mathbf{S}_-^\top\boldsymbol{E}_t\rangle \\
&= \langle \boldsymbol{\beta}_{t-1} - 2\boldsymbol{\beta}_t + \frac{\rho}{2}\mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}), \frac{2}{\rho}(\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}) + \mathbf{S}_-^\top\boldsymbol{E}_t\rangle \qquad (3.41) \\
&\overset{(c)}{=} -\frac{2}{\rho}\langle \boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}, \boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\rangle - \frac{2}{\rho}\langle \boldsymbol{\beta}_t, \boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\rangle + \langle \boldsymbol{\beta}_{t-1} - \boldsymbol{\beta}_t, \mathbf{S}_-^\top\boldsymbol{E}_t\rangle \\
&\quad - \langle \boldsymbol{\beta}_t, \mathbf{S}_-^\top\boldsymbol{E}_t\rangle + \langle \mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}), \boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\rangle + \frac{\rho}{2}\langle \mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}), \mathbf{S}_-^\top\boldsymbol{E}_t\rangle \\
&= -\frac{2}{\rho}\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\|_F^2 - \frac{2}{\rho}\|\boldsymbol{\beta}_t\|_F^2 + \frac{2}{\rho}\langle \boldsymbol{\beta}_t, \boldsymbol{\beta}_{t-1}\rangle + \langle \boldsymbol{\beta}_{t-1} - \boldsymbol{\beta}_t, \mathbf{S}_-^\top\boldsymbol{E}_t\rangle \\
&\quad - \langle \boldsymbol{\beta}_t, \mathbf{S}_-^\top\boldsymbol{E}_t\rangle + \langle \mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}), \boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\rangle + \frac{\rho}{2}\langle \mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}), \mathbf{S}_-^\top\boldsymbol{E}_t\rangle,
\end{aligned}
$$

where (a) comes from the KKT condition (3.36b), (b) and (c) come from (3.34).

We then utilize Young's inequality to bound the inner product terms in (3.41), which are

$$\frac{2}{\rho}\langle \boldsymbol{\beta}_t, \boldsymbol{\beta}_{t-1}\rangle \le \frac{2}{\rho}(\frac{1}{2\eta_1}\|\boldsymbol{\beta}_t\|_F^2 + \frac{\eta_1}{2}\|\boldsymbol{\beta}_{t-1}\|_F^2) = \frac{1}{\rho\eta_1}\|\boldsymbol{\beta}_t\|_F^2 + \frac{\eta_1}{\rho}\|\boldsymbol{\beta}_{t-1}\|_F^2, \qquad (3.42)$$

$$\langle \boldsymbol{\beta}_{t-1} - \boldsymbol{\beta}_t, \mathbf{S}_-^\top\boldsymbol{E}_t\rangle \le \frac{1}{2\eta_2}\|\boldsymbol{\beta}_{t-1} - \boldsymbol{\beta}_t\|_F^2 + \frac{\eta_2}{2}\|\mathbf{S}_-^\top\boldsymbol{E}_t\|_F^2, \qquad (3.43)$$

$$-\langle \boldsymbol{\beta}_t, \mathbf{S}_-^\top\boldsymbol{E}_t\rangle \le \frac{1}{2\eta_3}\|\boldsymbol{\beta}_t\|_F^2 + \frac{\eta_3}{2}\|\mathbf{S}_-^\top\boldsymbol{E}_t\|_F^2, \qquad (3.44)$$

$$\langle \mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}), \boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\rangle \le \frac{1}{2\eta_4}\|\mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1})\|_F^2 + \frac{\eta_4}{2}\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\|_F^2, \qquad (3.45)$$

$$\frac{\rho}{2}\langle \mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}), \mathbf{S}_-^\top\boldsymbol{E}_t\rangle \le \frac{\rho}{4\eta_5}\|\mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1})\|_F^2 + \frac{\rho\eta_5}{4}\|\mathbf{S}_-^\top\boldsymbol{E}_t\|_F^2, \qquad (3.46)$$

where $\eta_1, \eta_2, \eta_3, \eta_4, \eta_5$ are any positive constants.

Substitute (3.42) - (3.46) into (3.41) gives

$$
\begin{aligned}
&\langle \frac{\rho}{2}\mathbf{S}_-\mathbf{S}_-^\top(\boldsymbol{E}_t - \boldsymbol{E}_{t+1}) - \mathbf{S}_-\boldsymbol{\beta}_{t+1}, \boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star\rangle \\
&\leq (-\frac{2}{\rho} + \frac{1}{2\eta_2} + \frac{\eta_4}{2})\|\boldsymbol{\beta}_t - \boldsymbol{\beta}_{t-1}\|_F^2 + (-\frac{2}{\rho} + \frac{1}{\rho\eta_1} + \frac{1}{2\eta_3})\|\boldsymbol{\beta}_t\|_F^2 + \frac{\eta_1}{\rho}\|\boldsymbol{\beta}_{t-1}\|_F^2 \\
&\quad + (\frac{1}{2\eta_4} + \frac{\rho}{4\eta_5})\|\mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1})\|_F^2 + (\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\|\mathbf{S}_-^\top\boldsymbol{E}_t\|_F^2 \\
&= (-\frac{2}{\rho} + \frac{1}{2\eta_2} + \frac{\eta_4}{2})(\|\boldsymbol{\beta}_t\|_F^2 + \|\boldsymbol{\beta}_{t-1}\|_F^2 - 2\langle\boldsymbol{\beta}_t, \boldsymbol{\beta}_{t-1}\rangle) + (-\frac{2}{\rho} + \frac{1}{\rho\eta_1} + \frac{1}{2\eta_3})\|\boldsymbol{\beta}_t\|_F^2 + \frac{\eta_1}{\rho}\|\boldsymbol{\beta}_{t-1}\|_F^2 \\
&\quad + (\frac{1}{2\eta_4} + \frac{\rho}{4\eta_5})\|\mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1})\|_F^2 + (\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\|\mathbf{S}_-^\top\boldsymbol{E}_t\|_F^2 \\
&= (-\frac{4}{\rho} + \frac{1}{2\eta_2} + \frac{\eta_4}{2} + \frac{1}{\rho\eta_1} + \frac{1}{2\eta_3})\|\boldsymbol{\beta}_t\|_F^2 + (-\frac{2}{\rho} + \frac{1}{2\eta_2} + \frac{\eta_4}{2} + \frac{\eta_1}{\rho})\|\boldsymbol{\beta}_{t-1}\|_F^2 \\
&\quad + (\frac{1}{2\eta_4} + \frac{\rho}{4\eta_5})\|\mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1})\|_F^2 + (\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\|\mathbf{S}_-^\top\boldsymbol{E}_t\|_F^2 + (\frac{2}{\rho} - \frac{1}{2\eta_2} - \frac{\eta_4}{2})\langle\boldsymbol{\beta}_t, \boldsymbol{\beta}_{t-1}\rangle \\
&\leq (-\frac{4}{\rho} + \frac{1}{2\eta_2} + \frac{\eta_4}{2} + \frac{1}{\rho\eta_1} + \frac{1}{2\eta_3} + \frac{2}{\rho\eta_6} - \frac{1}{2\eta_2\eta_6} - \frac{\eta_4}{2\eta_6})\|\boldsymbol{\beta}_t\|_F^2 \\
&\quad + (-\frac{2}{\rho} + \frac{1}{2\eta_2} + \frac{\eta_4}{2} + \frac{\eta_1}{\rho} + \frac{2\eta_6}{\rho} - \frac{\eta_6}{2\eta_2} - \frac{\eta_4\eta_6}{2})\|\boldsymbol{\beta}_{t-1}\|_F^2 \\
&\quad + (\frac{1}{2\eta_4} + \frac{\rho}{4\eta_5})\|\mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1})\|_F^2 + (\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\|\mathbf{S}_-^\top\boldsymbol{E}_t\|_F^2.
\end{aligned}
\tag{3.47}
$$

With (3.47) and (3.40), we obtain an upper bound for (3.39), which is

$$
\begin{aligned}
&\mathcal{L}_t(\boldsymbol{\Theta}_t) - \mathcal{L}_t(\boldsymbol{\Theta}^\star) \\
&\leq \frac{\sigma_{\max}(\eta_t \mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2}\Big(\|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star\|_F^2 - \|\boldsymbol{\Theta}_{t+1} - \boldsymbol{\Theta}^\star\|_F^2\Big) \\
&\quad + \frac{\sigma_{\max}(\eta_t \mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2}\|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}\|_F^2 \\
&\quad + (\frac{1}{2\eta_2} - \frac{4}{\rho} + \frac{\eta_4}{2} + \frac{1}{\rho\eta_1} + \frac{1}{2\eta_3} + \frac{2}{\rho\eta_6} - \frac{1}{2\eta_2\eta_6} - \frac{\eta_4}{2\eta_6})\|\boldsymbol{\beta}_t\|_F^2 \\
&\quad + (\frac{\eta_1}{\rho} - \frac{2}{\rho} + \frac{1}{2\eta_2} + \frac{\eta_4}{2} + \frac{2\eta_6}{\rho} - \frac{\eta_6}{2\eta_2} - \frac{\eta_4\eta_6}{2})\|\boldsymbol{\beta}_{t-1}\|_F^2 \\
&\quad + (\frac{1}{2\eta_4} + \frac{\rho}{4\eta_5})\|\mathbf{S}_-^\top(\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1})\|_F^2 + (\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\|\mathbf{S}_-^\top \boldsymbol{E}_t\|_F^2 \qquad (3.48) \\
&\leq \frac{\sigma_{\max}(\eta_t \mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2}\Big(\|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star\|_F^2 - \|\boldsymbol{\Theta}_{t+1} - \boldsymbol{\Theta}^\star\|_F^2\Big) \\
&\quad + (\frac{1}{2\eta_2} - \frac{4}{\rho} + \frac{\eta_4}{2} + \frac{1}{\rho\eta_1} + \frac{1}{2\eta_3} + \frac{2}{\rho\eta_6} - \frac{1}{2\eta_2\eta_6} - \frac{\eta_4}{2\eta_6})\|\boldsymbol{\beta}_t\|_F^2 \\
&\quad + (\frac{\eta_1}{\rho} - \frac{2}{\rho} + \frac{1}{2\eta_2} + \frac{\eta_4}{2} + \frac{2\eta_6}{\rho} - \frac{\eta_6}{2\eta_2} - \frac{\eta_4\eta_6}{2})\|\boldsymbol{\beta}_{t-1}\|_F^2 \\
&\quad + (\frac{\sigma_{\max}(\eta_t \mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2} + \frac{\sigma_{\max}^2(\mathbf{S}_-)}{2\eta_4} + \frac{\rho\sigma_{\max}^2(\mathbf{S}_-)}{4\eta_5})\|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}\|_F^2 \\
&\quad + (\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\sigma_{\max}^2(\mathbf{S}_-)\|\boldsymbol{E}_t\|_F^2.
\end{aligned}
$$

We then utilize (3.35) to rewrite $\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}$ as

$$
\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1} = (\eta_t \mathbf{I} + 2\rho\mathbf{D})^{-1}\Big(\partial\mathcal{L}_t(\boldsymbol{\Theta}_t) + 2\mathbf{S}_-\boldsymbol{\beta}_t - \mathbf{S}_-\boldsymbol{\beta}_{t-1}\Big), \qquad (3.49)
$$

and bound $\|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}\|_F^2$ as

$$\begin{aligned}\|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_{t+1}\|_F^2 &= \|(\eta_t\mathbf{I} + 2\rho\mathbf{D})^{-1}\Big(\partial\mathcal{L}_t(\boldsymbol{\Theta}_t) + 2\mathbf{S}_-\boldsymbol{\beta}_t - \mathbf{S}_-\boldsymbol{\beta}_{t-1}\Big)\|_F^2 \\ &\leq \frac{1}{\sigma_{\min}^2(\eta_t\mathbf{I} + 2\rho\mathbf{D})}\|\partial\mathcal{L}_t(\boldsymbol{\Theta}_t)\|_F^2 + \frac{4\sigma_{\max}^2(\mathbf{S}_-)}{\sigma_{\min}^2(\eta_t\mathbf{I} + 2\rho\mathbf{D})}\|\boldsymbol{\beta}_t\|_F^2 \\ &\quad + \frac{\sigma_{\max}^2(\mathbf{S}_-)}{\sigma_{\min}^2(\eta_t\mathbf{I} + 2\rho\mathbf{D})}\|\boldsymbol{\beta}_{t-1}\|_F^2,\end{aligned} \tag{3.50}$$

where $\sigma_{\min}(\eta_t\mathbf{I} + 2\rho\mathbf{D})$ is the lower bound of the nonzero singular values of $\eta_t\mathbf{I} + 2\rho\mathbf{D}$.

Substitute (3.50) into (3.48) we obtain

$$\begin{aligned}\mathcal{L}_t(\boldsymbol{\Theta}_t) - \mathcal{L}_t(\boldsymbol{\Theta}^\star) &\leq \frac{\sigma_{\max}(\eta_t\mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2}\Big(\|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star\|_F^2 - \|\boldsymbol{\Theta}_{t+1} - \boldsymbol{\Theta}^\star\|_F^2\Big) \\ &\quad + (c_1 + 4c_{\mathcal{N}})\|\boldsymbol{\beta}_t\|_F^2 + (c_2 + c_{\mathcal{N}})\|\boldsymbol{\beta}_{t-1}\|_F^2 \\ &\quad + \frac{c_{\mathcal{N}}}{\sigma_{\max}^2(\mathbf{S}_-)}\|\partial\mathcal{L}_t(\boldsymbol{\Theta}_t)\|_F^2 + (\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\sigma_{\max}^2(\mathbf{S}_-)\|\boldsymbol{E}_t\|_F^2,\end{aligned} \tag{3.51}$$

where c_1, c_2 and $c_{\mathcal{N}}$ are defined as follows:

$$\begin{aligned}c_1 &:= \frac{1}{2\eta_2} - \frac{4}{\rho} + \frac{\eta_4}{2} + \frac{1}{\rho\eta_1} + \frac{1}{2\eta_3} + \frac{2}{\rho\eta_6} - \frac{1}{2\eta_2\eta_6} - \frac{\eta_4}{2\eta_6}, \\ c_2 &:= \frac{\eta_1}{\rho} - \frac{2}{\rho} + \frac{1}{2\eta_2} + \frac{\eta_4}{2} + \frac{2\eta_6}{\rho} - \frac{\eta_6}{2\eta_2} - \frac{\eta_4\eta_6}{2}, \\ c_{\mathcal{N}} &:= \Big(\frac{\sigma_{\max}(\eta_t\mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2} + \frac{\sigma_{\max}^2(\mathbf{S}_-)}{2\eta_4} + \frac{\rho\sigma_{\max}^2(\mathbf{S}_-)}{4\eta_5}\Big)\frac{\sigma_{\max}^2(\mathbf{S}_-)}{\sigma_{\min}^2(\eta_t\mathbf{I} + 2\rho\mathbf{D})}.\end{aligned}$$

Carefully choose $\eta_1, \eta_2, \eta_3, \eta_4, \eta_5$, and η_6, we can make $c_1 + 4c_{\mathcal{N}} = -(c_2 + c_{\mathcal{N}}) = -c$, where

$c > 0$. Then (3.51) can be further simplified as

$$\begin{aligned}\mathcal{L}_t(\boldsymbol{\Theta}_t) - \mathcal{L}_t(\boldsymbol{\Theta}^\star) \leq & \frac{\sigma_{\max}(\eta_t \mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2}\Big(\|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}^\star\|_F^2 - \|\boldsymbol{\Theta}_{t+1} - \boldsymbol{\Theta}^\star\|_F^2\Big) \\ & + \frac{c_{\mathcal{N}}}{\sigma_{\max}^2(\mathbf{S}_-)}\|\partial \mathcal{L}_t(\boldsymbol{\Theta}_t)\|_F^2 + c(\|\boldsymbol{\beta}_{t-1}\|_F^2 - \|\boldsymbol{\beta}_t\|_F^2) \\ & + (\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\sigma_{\max}^2(\mathbf{S}_-)\|\boldsymbol{E}_t\|_F^2.\end{aligned} \tag{3.52}$$

To satisfy $c_1 + 4c_{\mathcal{N}} = -(c_2 + c_{\mathcal{N}})$, one example is to set $\eta_4 = \frac{2\eta_6}{(\eta_6-1)^2}(\frac{1}{\rho}(\eta_1 + \frac{1}{\eta_1} + \frac{2}{\eta_6} + 2\eta_6) + \frac{1}{2\eta_2}(2 - \eta_6 - \frac{1}{\eta_6}) + \frac{1}{2\eta_3} + 5c_{\mathcal{N}})$.

Summarizing both sides of (3.52) from $t = 1$ to $t = T$ leads to the accumulated network regret $\mathbf{Reg}_T^S$:

$$\begin{aligned}\mathbf{Reg}_T^S \leq & \frac{\sigma_{\max}(\eta_t \mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2}\Big(\|\boldsymbol{\Theta}_1 - \boldsymbol{\Theta}^\star\|_F^2 - \|\boldsymbol{\Theta}_{T+1} - \boldsymbol{\Theta}^\star\|_F^2\Big) + c(\|\boldsymbol{\beta}_0\|_F^2 - \|\boldsymbol{\beta}_T\|_F^2) \\ & + \sum_{t=1}^T \frac{c_{\mathcal{N}}}{\sigma_{\max}^2(\mathbf{S}_-)}\|\partial \mathcal{L}_t(\boldsymbol{\Theta}_t)\|_F^2 + \sum_{t=1}^T(\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\sigma_{\max}^2(\mathbf{S}_-)\|\boldsymbol{E}_t\|_F^2 \\ \leq & \frac{\sigma_{\max}(\eta_t \mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2}\|\boldsymbol{\Theta}_1 - \boldsymbol{\Theta}^\star\|_F^2 + c\|\boldsymbol{\beta}_0\|_F^2 + \sum_{t=1}^T \frac{c_{\mathcal{N}}}{\sigma_{\max}^2(\mathbf{S}_-)}\|\partial \mathcal{L}_t(\boldsymbol{\Theta}_t)\|_F^2 \\ & + \sum_{t=1}^T(\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\sigma_{\max}^2(\mathbf{S}_-)\|\boldsymbol{E}_t\|_F^2 \\ = & \frac{\sigma_{\max}(\eta_t \mathbf{I} + \frac{\rho}{2}\mathbf{S}_+\mathbf{S}_+^\top)}{2}\|\boldsymbol{\Theta}^\star\|_F^2 + \sum_{t=1}^T \frac{c_{\mathcal{N}}}{\sigma_{\max}^2(\mathbf{S}_-)}\|\partial \mathcal{L}_t(\boldsymbol{\Theta}_t)\|_F^2 \\ & + \sum_{t=1}^T(\frac{\eta_2}{2} + \frac{\eta_3}{2} + \frac{\rho\eta_5}{4})\sigma_{\max}^2(\mathbf{S}_-)\|\boldsymbol{E}_t\|_F^2.\end{aligned} \tag{3.53}$$

The last equality comes from the initialization that $\boldsymbol{\Theta}_1 = \mathbf{0}$ and $\boldsymbol{\beta}_1 = \mathbf{0}$ and thus $\boldsymbol{\beta}_0 = \mathbf{0}$.

Assumption 5 assumes that $\|\boldsymbol{\theta}^\star\|_F \leq C_\theta$, which implies that $\|\boldsymbol{\Theta}^\star\|_F \leq \sqrt{N}C_\theta$. Assumption 4 assumes that $\|\partial\mathcal{L}_{i,t}(\boldsymbol{\theta}_t)\|_F \leq G$, which indicates that the aggregated cost functions $\mathcal{L}_t$ satisfy $\|\partial\mathcal{L}_t(\boldsymbol{\theta})\|_2 \leq \sqrt{N}G$. Setting $\rho = \eta_t = \eta_2 = \eta_3 = 1/\mathcal{O}(\sqrt{T})$, QC-ODKLA achieves the sublinear regret:

$$\mathbf{Reg}_T^S \leq (\sqrt{N}C_\theta + \frac{\sqrt{N}G}{\sigma_{\max}^2(\mathbf{S}_-)} + \sigma_{\max}^2(\mathbf{S}_-)\zeta)\mathcal{O}(\sqrt{T}), \tag{3.54}$$

where $\zeta := \max\{\sqrt{N}\varrho\mu, \sqrt{2NL}\Delta/2\}$, with ϱ and μ being the predefined censoring threshold parameters and Δ being the length of the quantization interval. ■

3.7.2 Datasets descriptions

The detailed descriptions of the six datasets are listed below.

Tom's hardware. This dataset contains $T_{total} = 9725$ samples with $\mathbf{x}_t \in \mathbb{R}^{96}$ including the number of created discussions and authors interacting of a topic and $y_t \in \mathbb{R}$ representing the average number of displays to a visitor about that topic [87].

Twitter. This dataset consists of $T_{total} = 98700$ samples with $\mathbf{x}_t \in \mathbb{R}^{77}$ being a feature vector reflecting the number of new interactive authors and the length of discussions on a given topic, etc., and $y_t \in \mathbb{R}$ representing the average number of active discussions on a certain topic. The learning task is to predict the popularity of these topics [87].

Energy. This dataset contains $T_{total} = 18600$ samples with $\mathbf{x}_t \in \mathbb{R}^{28}$ describing the humidity and temperature in different areas of the house, pressure, wind speed, and viability outside, while $y_t \in \mathbb{R}$ denoting the total energy consumption in the house [88].

Air quality. This dataset contains $T_{total} = 7320$ samples measured by a gas multi-sensor device in an Italian city, where $\mathbf{x}_t \in \mathbb{R}^{13}$ represents the hourly concentration of CO, NOx, NO2, etc and $y_t \in \mathbb{R}$ denotes the concentration of polluting chemicals in the air [89].

Conductivity: This dataset contains $T_{total} = 21260$ samples extracted from superconductors, where $\mathbf{x}_t \in \mathbb{R}^{81}$ represents critical information to construct superconductor such as

density and mass of atoms. The task is to predict the critical temperature $y_t \in \mathbb{R}$ which creates a superconductor [116].

Blood data: This dataset contains $T_{total} = 61000$ samples recorded by patient monitors at different hospitals where $\mathbf{x}_t \in \mathbb{R}^2$ and the goal is to predict the blood pressure $y_t \in \mathbb{R}$ based on several physiological parameters from Photoplethysmography and Electrocardiogram signals [117].

Chapter 4: Communication-Efficient Decentralized Dynamic Kernel Learning

4.1 Introduction

Online learning problems where data arrive in a streaming manner have been considered in Chapter 3, with kernel-based algorithms developed to tackle them in a communication and computation efficient way. In this chapter, we further consider cases where the dynamics underlying the streaming data is varying, where the optimal function to be estimated could be the state in brain graphs, the temporal process propagating over a time-varying network, or human's activity in motion [118]. In these cases, the function of interest itself may vary over time with unknown dynamics, in addition to the evolving data. Thus, no single policy is always good, and thus minimizing the static regret against a fixed policy is no longer suitable. It is because the static regret is the difference between the cumulative loss caused by the learning algorithm and that caused by the best static model in hindsight. Meanwhile, since Chapter 3 assumes that the sampling distribution of the streaming data is stationary, the optimization problem therein enforces consensus between the current state and its previous state, which may not be suitable when the function of interest varies over time. In this chapter, instead of assuming the streaming data come from a stationary distribution, we aim to solve the online learning problem with unknown dynamics.

Online (convex) learning/optimization problem in dynamic non-stationary environments was investigated in [119, 120], where instead of assuming there exists an optimal deterministic action in hindsight, a sequence of optimal actions (minimizers) corresponding to a sequence of cost functions are taken to minimize a *dynamic* regret. For a single agent case, the dynamic regret is the cost accumulation of an action $\mathbf{a}_t$ as compared with a best action

$\mathbf{a}_t^*$ at each time:

$$\mathbf{Reg}_T^D = \sum_{t=1}^{T} \mathcal{L}_t(\mathbf{a}_t) - \sum_{t=1}^{T} \mathcal{L}_t(\mathbf{a}_t^*), \tag{4.1}$$

where $\mathbf{a}_t^* = \arg\min_{\mathbf{a}_t \in \mathbb{R}^p} \mathcal{L}_t(\mathbf{a}_t)$, and actions $\mathbf{a}_t$ are usually assumed to lie in a Euclidean space $\mathbb{R}^p$ in standard online convex optimization problems. As in the static case, online convex optimization concerns designing methods such that $\mathbf{Reg}_T^D$ grows sublinearly in T [121–126]. Distributed online convex learning has also been investigated [101, 103, 127–129]. However, all these methods assume that the best decisions lie in the Euclidean space, while nonlinear function learning in environments with unknown dynamics remains a largely exploited territory.

In this chapter, we continue our research on utilizing kernels for nonlinear function estimation in a dynamic environment to leverage the representation power of nonparametric kernel methods. More specifically, we focus on dynamic kernel learning over a decentralized network, where all agents receive online streaming data collected from non-stationary environments. Notice that [130, 131] also investigated the dynamic kernel learning problem in non-stationary environments; however, they all focus on the centralized case and utilize gradient descent based method for optimization. The curse of dimensionality issue of online kernel learning is circumvented by random feature (RF) mapping in [130] and by restricting the number of function parameters in [131]. We also utilize RF mapping to alleviate the computational complexity issue in decentralized dynamic kernel learning as in [130]. However, the significance of RF mapping is beyond reducing computational complexity, it also enables the formulation of the decentralized dynamic kernel learning problem as a decentralized dynamic convex optimization problem. Instead of the gradient descent method, we utilize the alternating direction method of multipliers (ADMM) for distributed implementation. We further add the communication censoring and quantization strategies to save communication resources. Notice that [128] utilizes the "event-triggered" rule in decentralized online convex optimization with unknown dynamics to save communication resources,

where the design of their events shares similarities with our communication-censoring rules. However, they assume that the best decisions lie in the Euclidean space, and thus are not the same as our kernel methods.

This work lies in the intersection of (distributed) online convex optimization in non-stationary environments, scalable online kernel learning, and efficient communication-computation schemes. Related literature in (distributed) online convex optimization in non-stationary environments has been reviewed above. The related literature in scalable online kernel learning and efficient communication-computation schemes is referred to the reviews in Chapter 2 and Chapter 3.

Relative to prior art, our contributions are summarized as follows.

- We formulate the decentralized dynamic kernel learning problem as a decentralized convex optimization problem and solve it under the consensus optimization framework using ADMM. To the best of our knowledge, this is the first work that solves the decentralized dynamic kernel learning problem. The key of our solution is to apply RF mapping, which not only circumvents the curse of dimensionality issue of conventional kernel methods but also enables consensus on a set of model parameters of fixed size in the RF space.

- Utilizing both communication-censoring and quantization strategies, we develop the quantized and communication-censored decentralized dynamic kernel learning via ADMM (QC-DDKL) algorithm, which achieves desired learning performance given limited communication resources and energy supply.

- In addition, we analyze the dynamic regret bound of QC-ODKLA. We show that by carefully tuning the communication censoring parameters such that the censoring thresholds converge to zero asymptotically in time, QC-DDKL achieves sublinear regret $\mathcal{O}(\sqrt{T})$ over T time slots under mild conditions, which is on the same order as that of the state-of-the-art online algorithms for the static case with full communication.

- Finally, we test the performance of our proposed QC-DDKL algorithm on both synthetic and real datasets. The results corroborate that QC-DDKL exhibits attractive learning performance and communication efficiency. Such salient features make it an attractive solution for broad applications where decentralized learning from streaming data with unknown dynamics is at its core.

Organization. The remaining of this chapter is organized as follows. Section 4.2 formulates the decentralized dynamic kernel learning problem and Section 4.3 develops the quantized and communication-censored decentralized dynamic kernel learning algorithm. Section 4.4 presents the theoretical results and Section 4.5 reports the numerical tests using real datasets. Concluding remarks are summarized in Section 4.6.

4.2 Problem Statement

In this chapter, we consider the same network model and communication model described in Section 1.2. The network with N agents and r arcs is bidirectionally connected with an underlying undirected communication graph denoted as $\mathcal{G} = (\mathcal{N}, \mathcal{A})$. Here the cardinality of $\mathcal{N}$ and $\mathcal{A}$ are N and $2r$, respectively. Two agents i and j are called as neighbors when $(i, j) \in \mathcal{A}$ and, by the symmetry of the network, $(j, i) \in \mathcal{A}$. For agent i, its neighbors within a one-hop communication range are in the set $\mathcal{N}_i = \{j|(j, i) \in \mathcal{A}\}$. The cardinality $|\mathcal{N}_i|$ is also known as the degree d_i of agent i and the degree matrix of the communication graph is $\mathbf{D} \in \mathbb{R}^{N\times N}$ whose ith diagonal element is $d_i, \forall i$. Define the symmetric adjacency matrix associated with the communication graph as $\mathbf{W} \in \mathbb{R}^{N\times N}$ whose (i, j)th entry is 1 if agent i and j are neighbors or 0 otherwise. Define the unsigned incidence matrix and the signed incidence matrix of the communication graph as $\mathbf{S}_+ \in \mathbb{R}^{N\times 2r}$ and $\mathbf{S}_- \in \mathbb{R}^{N\times 2r}$, respectively. Equations (3.31) associated with this network also apply in this chapter, and

we repeat them here for reference:

$$\begin{aligned}\mathbf{D}+\mathbf{W}&=\frac{1}{2}\mathbf{S}_{+}\mathbf{S}_{+}^{\top},\\ \mathbf{D}-\mathbf{W}&=\frac{1}{2}\mathbf{S}_{-}\mathbf{S}_{-}^{\top}.\end{aligned}\tag{4.2}$$

Consider the case that each agent in the network only has access to its locally observed sequential data $\{\mathbf{x}_{i,t}, y_{i,t}\}$, with $\mathbf{x}_{i,t} \in \mathbb{R}^d$ and $y_{i,t} \in \mathbb{R}$. In a decentralized setting with unknown dynamics and privacy concerns, agents seek to collaboratively find the best function $f_t, t \in [T]$ "on the fly" and without exchanging their raw data such that $y_{i,t} = f_t(\mathbf{x}_{i,t}) + e_{i,t}$ for $\{\{\mathbf{x}_{i,t}, y_{i,t}\}\}_{i=1}^{N}$, where $e_{i,t}$ is minimized accordingly to certain optimality metric. The dynamic regret in this case (4.1) is modified to

$$\mathbf{Reg}_T^D = \sum_{i=1}^{N}\sum_{t=1}^{T}\left(\mathcal{L}_{i,t}(f_t(\mathbf{x}_{i,t}), y_{i,t}) - \mathcal{L}_{i,t}(f_t^*(\mathbf{x}_{i,t}), y_{i,t})\right), \tag{4.3}$$

where $\mathcal{L}_{i,t}(\cdot,\cdot)$ represents the cost measure at agent i at time t, and $f_t^* = \arg\min_f \sum_{i=1}^{N} \mathcal{L}_{i,t}$ is the best network action at time t.

Suppose that the function of interest f_t belongs to the reproducing kernel Hilbert space (RKHS) $\mathcal{H} := \{f | f(\mathbf{x}) = \sum_{t=1}^{\infty} \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t)\}$ induced by a positive semidefinite kernel $\kappa(\mathbf{x}, \mathbf{x}_t) : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ that measures the similarity between $\mathbf{x}$ and $\mathbf{x}_t$, for all $\mathbf{x}, \mathbf{x}_t \in \mathcal{X}$. Due to the nature of streaming data, f_t is estimated using the received data $\{\{(\mathbf{x}_{i,\tau}, y_{i,\tau})\}_{\tau=1}^{t-1}\}_{i=1}^{N}$ up till time t. When all data are centrally available, the Representer theorem indicates that an estimate of the optimal function f_t admits

$$\hat{f}_t(\mathbf{x}) = \sum_{i=1}^{N}\sum_{\tau=1}^{t-1} \alpha_{i,\tau}\kappa(\mathbf{x}_{i,\tau}, \mathbf{x}), \tag{4.4}$$

where $\{\{\alpha_{i,\tau}\}_{\tau=1}^{t-1}\}_{i=1}^{N}$ are coefficients to be learned at each time instant.

It is clear that learning $\alpha_{i,\tau}$'s becomes more and more computationally intense when the number of streaming data increases. In addition, in a decentralized setting, agents need to communicate $\alpha_{i,\tau}$'s and their raw data $\mathbf{x}_{i,t}$ with their neighbors to facilitate collaborative learning, which not only incurs high communication costs but also violates the privacy concern. RF mapping is thus utilized to tackle these challenges, which approximates (4.4) as

$$\hat{f}_t(\mathbf{x}) = \boldsymbol{\theta}_t^\top \boldsymbol{\phi}_L(\mathbf{x}), \tag{4.5}$$

where $\boldsymbol{\theta} \in \mathbb{R}^{2L}$ is the new decision vector to be learned in the RF space, and $\boldsymbol{\phi}_L(\mathbf{x})$ is the RF-mapped data given by

$$\boldsymbol{\phi}_L(\mathbf{x}) := \sqrt{\frac{1}{L}}[\phi(\mathbf{x}, \boldsymbol{\omega}_1), \ldots, \phi(\mathbf{x}, \boldsymbol{\omega}_L)]^\top, \tag{4.6}$$

with $\phi(\mathbf{x}, \boldsymbol{\omega}_l)$ defined as

$$\phi(\mathbf{x}, \boldsymbol{\omega}_l) = [\cos(\boldsymbol{\omega}_l^\top \mathbf{x}), \sin(\boldsymbol{\omega}_l^\top \mathbf{x})]^\top, \tag{4.7}$$

and $\{\boldsymbol{\omega}_l\}_{l=1}^L$ are randomly drawn from $p_\kappa(\boldsymbol{\omega})$, which is the inverse Fourier transform of κ. For a Gaussian kernel $\kappa(\mathbf{x}_t, \mathbf{x}_\tau) = \exp(-\|\mathbf{x}_\mathrm{t} - \mathbf{x}_\tau\|_2^2/(2\sigma^2))$, we have $p_\kappa(\boldsymbol{\omega}) \sim \mathcal{N}(\mathbf{0}, \sigma^{-2}\mathbf{I})$. In this way, the dynamic regret (4.3) is customized to

$$\mathbf{Reg}_T^D = \sum_{i=1}^N \sum_{t=1}^T (\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}; \mathcal{S}_{i,t}) - \mathcal{L}_{i,t}(\boldsymbol{\theta}_t^*; \mathcal{S}_{i,t})), \tag{4.8}$$

where $\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}; \mathcal{S}_{i,t}) := \mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}^\top \boldsymbol{\phi}_L(\mathbf{x}_{i,t}), y_{i,t})$, $\mathcal{S}_{i,t} = (\mathbf{x}_{i,t}, y_{i,t})$, and $\boldsymbol{\theta}_{i,t}$ is agent i's local copy of $\boldsymbol{\theta}_t$.

With data coming in a streaming manner and function estimation operated in an online fashion, the decentralized dynamic kernel learning problem then turns to be the following

optimization problem:

$$\{\boldsymbol{\theta}_{i,t+1}\}_{i=1}^{N} := \underset{\{\boldsymbol{\theta}_i\}_{i=1}^{N}}{\operatorname{argmin}} \quad \sum_{i=1}^{N} \mathcal{L}_{i,t}(\boldsymbol{\theta}_i; \mathcal{S}_{i,t}) \\ \text{s.t.} \qquad \boldsymbol{\theta}_i = \boldsymbol{\theta}_j, \qquad \forall (i,j) \in \mathcal{A}. \tag{4.9}$$

At time t, each agent i receives a new data pair $\mathcal{S}_{i,t}$, which incurs a local cost $\mathcal{L}_{i,t}$. Note that, this cost depends on the new data only. Together with the current local copy $\boldsymbol{\theta}_{i,t}$ of $\boldsymbol{\theta}_t$, the local cost $\mathcal{L}_{i,t}$, and information from their neighbors, i.e., $\boldsymbol{\theta}_{j,t}, \forall j \in \mathcal{N}_i$, agent i then updates the estimate of $\boldsymbol{\theta}_{i,t+1}$. In the next section, we first propose an ADMM-based decentralized dynamic kernel learning algorithm to show how $\boldsymbol{\theta}_{i,t+1}, \forall i$, are estimated, then we utilize the communication-censoring and quantization strategies to improve the communication efficiency of the proposed algorithm.

4.3 Algorithm Development

In this section, we first utilize ADMM to solve (4.9) and then add the communication-censoring and quantization techniques to develop a communication efficient solution to decentralized dynamic kernel learning.

To utilize ADMM, we first introduce an auxiliary variables $\boldsymbol{z}_{ij}$ to enforce consensus constraints such that $\boldsymbol{\theta}_i = \boldsymbol{\theta}_j$ for all arcs $(i,j) \in \mathcal{A}$, which converts the optimization problem (4.9) into

$$\min_{\{\boldsymbol{\theta}_i, \boldsymbol{z}_{ij}\}_{i=1}^{N}} \quad \sum_{i=1}^{N} \mathcal{L}_{i,t}(\boldsymbol{\theta}_i; \mathcal{S}_{i,t}) \\ \text{s.t.} \qquad \boldsymbol{\theta}_i = \boldsymbol{z}_{ij}, \boldsymbol{\theta}_j = \boldsymbol{z}_{ij}, \qquad \forall (i,j) \in \mathcal{A}. \tag{4.10}$$

Note that $\{\boldsymbol{\theta}_i\}$ are separable when $\{\boldsymbol{z}_{ij}\}$ are fixed, and vice versa. The auxiliary variable $\boldsymbol{z}_{ij}$ can be written as a function of $\boldsymbol{\theta}_i$ and then be canceled out. Following [114], we develop

the dynamic decentralized kernel learning (DDKL) algorithm via ADMM where each agent updates its local primal variable $\boldsymbol{\theta}_i$ and local dual variable $\boldsymbol{\gamma}_i$ by

$$\boldsymbol{\theta}_{i,t+1} := \arg\min_{\boldsymbol{\theta}_i} \Big\{ \mathcal{L}_{i,t}(\boldsymbol{\theta}_i; \mathcal{S}_{i,t}) + \rho d_i \|\boldsymbol{\theta}_i\|_2^2 + \boldsymbol{\theta}_i^\top \Big[\boldsymbol{\gamma}_{i,t} - \rho \sum_{j \in \mathcal{N}_i} (\boldsymbol{\theta}_{i,t} + \boldsymbol{\theta}_{j,t}) \Big] \Big\}, \tag{4.11}$$

$$\boldsymbol{\gamma}_{i,t+1} = \boldsymbol{\gamma}_{i,t} + \rho \sum_{j \in \mathcal{N}_i} \Big(\boldsymbol{\theta}_{i,t+1} - \boldsymbol{\theta}_{j,t+1} \Big), \tag{4.12}$$

where d_i is the degree of agent i. Interested readers are referred to [114] for detailed derivation.

The iterative process (4.11) - (4.12) involves frequent communication, thus incurs high communication cost. To save communication resources, we utilize the communication-censoring strategy to avoid unnecessary communications and the quantization strategy to reduce the total number of bits transmitted in the learning process. The developed algorithm is called quantized and communication-censored decentralized dynamic kernel learning via ADMM (QC-DDKL).

The algorithm development process of QC-DDKL is the same as that of QC-ODKLA in Chapter 3.3.2. For clarity and consistency purposes, we briefly summarize the process. First, a new state variable $\hat{\boldsymbol{\theta}}_{i,t}$ is introduced for each agent i to record its latest broadcast primal variable up to time t. Then, the difference between agent i's updated state $\boldsymbol{\theta}_{i,t+1}$ and its previously transmitted state $\hat{\boldsymbol{\theta}}_{i,t}$ at time t is calculated:

$$\boldsymbol{h}_{i,t} = \boldsymbol{\theta}_{i,t+1} - \hat{\boldsymbol{\theta}}_{i,t}. \tag{4.13}$$

The magnitude of the state difference $\boldsymbol{h}_{i,t}$, an indicator of the informativeness of the local update, is assessed by an evaluation function

$$H_{i,t} = \|\boldsymbol{h}_{i,t}\|_2 - \varrho t^{-\mu}, \tag{4.14}$$

where $\varrho t^{-\mu}$ is a predefined time-dependent threshold function with $\varrho > 0$ and $\mu < 1$. If $H_{i,t} \geq 0$, then $\boldsymbol{\theta}_{i,t+1}$ is deemed informative, and agent i is allowed to transmit the quantized difference $Q(\boldsymbol{h}_{i,t})$ to its neighbors. Instead of quantizing and transmitting $\boldsymbol{\theta}_{i,t+1}$, we adopt the difference-based quantization scheme to facilitate the measurement and analysis of the impact of quantization [114]. Specifically, for each element $h_{i,t}^l (l = 1, \ldots, 2L)$ within the range of $[u, v)$, if we restrict the number of transmission bits to be b, then we can evenly divided the range $[u, v)$ to be $q = 2^b$ intervals of equal length $\Delta = (v - u)/q$. Then the rounding quantizer $Q(\cdot)$ applied to $h_{i,t}^l$ outputs

$$Q(h_{i,t}^l) = u + \left(\left\lfloor \frac{h_{i,t}^l - u}{\Delta} \right\rfloor + \frac{1}{2} \right) \Delta, \tag{4.15}$$

where $\lfloor \cdot \rfloor$ is the floor operation. In practice, it is not necessary to transmit $Q(h_{i,t}^l)$; instead, we can simply transmit the integer $k := \left\lfloor \frac{h_{i,t}^l - u}{\Delta} \right\rfloor$ using the b bits. Thus, the total number of bits for agent i to transmit the quantized difference $Q(\boldsymbol{h}_{i,t})$ to its neighbors is only $2Lb$ bits.

The whole communication process thus involves three parts: evaluation, quantization, and state update. If $H_{i,t} \geq 0$, then $\boldsymbol{\theta}_{i,t+1}$ is deemed informative, and agent i is allowed to transmit a quantized difference $Q(\boldsymbol{h}_{i,t})$ to its neighbors and updates its local state as $\hat{\boldsymbol{\theta}}_{i,t+1} = \hat{\boldsymbol{\theta}}_{i,t} + Q(\boldsymbol{h}_{i,t})$. Otherwise, $\boldsymbol{\theta}_{i,t+1}$ is censored, agent i sets $\hat{\boldsymbol{\theta}}_{i,t+1} = \hat{\boldsymbol{\theta}}_{i,t}$, and no information is transmitted. Similarly, upon receiving $Q(\boldsymbol{h}_{j,t})$ from its neighbor j, agent i updates the state variables of its neighbor's as $\hat{\boldsymbol{\theta}}_{j,t+1} = \hat{\boldsymbol{\theta}}_{j,t} + Q(\boldsymbol{h}_{j,t})$, otherwise, $\hat{\boldsymbol{\theta}}_{j,t+1} = \hat{\boldsymbol{\theta}}_{j,t}$.

With the communication censoring rule and the quantization scheme, the primal and

Algorithm 5 QC-DDKL (run at agent i)

Require: Kernel κ, hyper-parameters $(L, \rho, \eta_t, \varrho, \mu)$, initialize local variables to $\boldsymbol{\theta}_{i,1} = \mathbf{0}$, and $\boldsymbol{\gamma}_{i,1} = \mathbf{0}$, $\hat{\boldsymbol{\theta}}_{i,1} = Q(\boldsymbol{\theta}_{i,1})$, and $\hat{\boldsymbol{\theta}}_{j,1} = Q(\boldsymbol{\theta}_{j,1})$ for all $j \in \mathcal{N}_i$.

1: Draw L i.i.d. samples $\{\boldsymbol{\omega}_l\}_{l=1}^{L}$ from $p_\kappa(\boldsymbol{\omega})$ according to a common random seed.
2: **for** iterations $t = 1, 2, \ldots, T$ **do**
3: Receive a streaming data $(\mathbf{x}_{i,t}, y_{i,t})$.
4: Construct $\boldsymbol{\phi}(\mathbf{x}_{i,t})$ via (4.6).
5: Update local primal variable $\boldsymbol{\theta}_{i,t+1}$ by solving (4.16).
6: Calculate the difference $\boldsymbol{h}_{i,t}$ via (4.13) and quantize it as $Q(\boldsymbol{h}_{i,t})$ via (4.15).
7: If (4.14) is nonnegative, transmit $Q(\boldsymbol{h}_{i,t})$ to neighbors and set $\hat{\boldsymbol{\theta}}_{i,t+1} = \hat{\boldsymbol{\theta}}_{i,t} + Q(\boldsymbol{h}_{i,t})$. Else, set $\hat{\boldsymbol{\theta}}_{i,t+1} = \hat{\boldsymbol{\theta}}_{i,t}$ and do not transmit.
8: If receiving $Q(\boldsymbol{h}_{j,t})$ from neighbors j, update $\hat{\boldsymbol{\theta}}_{j,t+1} = \hat{\boldsymbol{\theta}}_{j,t} + Q(\boldsymbol{h}_{j,t})$. Else, set $\hat{\boldsymbol{\theta}}_{j,t+1} = \hat{\boldsymbol{\theta}}_{j,t}$.
9: Update local dual variable $\boldsymbol{\gamma}_{i,t+1}$ via (4.17).
10: **end for**

dual updates in (4.11) and (4.11) become

$$\boldsymbol{\theta}_{i,t+1} = \arg\min_{\boldsymbol{\theta}_i} \Big\{ \mathcal{L}_{i,t}(\boldsymbol{\theta}_i; \mathcal{S}_{i,t}) + \rho |\mathcal{N}_i| \|\boldsymbol{\theta}_i\|_2^2 + \boldsymbol{\theta}_i^\top \Big[\boldsymbol{\gamma}_{i,t} - \rho \sum_{j \in \mathcal{N}_i} (\hat{\boldsymbol{\theta}}_{i,t} + \hat{\boldsymbol{\theta}}_{j,t}) \Big] \Big\}, \tag{4.16}$$

$$\boldsymbol{\gamma}_{i,t+1} = \boldsymbol{\gamma}_{i,t} + \rho \sum_{j \in \mathcal{N}_i} \Big(\hat{\boldsymbol{\theta}}_{i,t+1} - \hat{\boldsymbol{\theta}}_{j,t+1} \Big). \tag{4.17}$$

The QC-DDKL algorithm is summarized Algorithm 5.

4.4 Regret Analysis

In this section, we analyze the dynamic regret bound of QC-DDKL, which is defined in (4.8). We prove that QC-DDKL achieves the optimal sublinear regret $\mathcal{O}(\sqrt{T})$ for convex local cost functions $\mathcal{L}_{i,t}$. The following commonly used assumptions are adopted.

Assumption 6. *The local cost functions $\mathcal{L}_{i,t}(\boldsymbol{\theta})$ are strongly convex with constants $m_{\mathcal{L}_{i,t}} > 0$ such that $\forall i \in \mathcal{N}$, $\langle \nabla \mathcal{L}_{i,t}(\tilde{\boldsymbol{\theta}}_a) - \nabla \mathcal{L}_{i,t}(\tilde{\boldsymbol{\theta}}_b), \tilde{\boldsymbol{\theta}}_a - \tilde{\boldsymbol{\theta}}_b \rangle \geq m_{\mathcal{L}_{i,t}} \|\tilde{\boldsymbol{\theta}}_a - \tilde{\boldsymbol{\theta}}_b\|_2^2$, for any $\tilde{\boldsymbol{\theta}}_a, \tilde{\boldsymbol{\theta}}_b \in \mathbb{R}^{2L}$. The minimum convexity constant is $m_{\mathcal{L}} := \inf_{i,t} m_{\mathcal{L}_{i,t}}$. The gradients of the local cost functions are Lipschitz continuous with constants $M_{\mathcal{L}_{i,t}} > 0, \forall\ i, t$. That is, $\|\nabla \mathcal{L}_{i,t}(\tilde{\boldsymbol{\theta}}_a) - \nabla \mathcal{L}_{i,t}(\tilde{\boldsymbol{\theta}}_b)\|_2 \leq M_{\mathcal{L}_{i,t}} \|\tilde{\boldsymbol{\theta}}_a - \tilde{\boldsymbol{\theta}}_b\|_2$ for any agent i given any $\tilde{\boldsymbol{\theta}}_a, \tilde{\boldsymbol{\theta}}_b \in \mathbb{R}^{2L}$. The maximum*

Lipschitz constant is $M_{\mathcal{L}} := \sup_{i,t} M_{\mathcal{L}_{i,t}}$.

Assumption 6 ensures that the aggregated objective functions $\mathcal{L}_t := \sum_{i=1}^{L} \mathcal{L}_{i,t}$ are strongly convex with constant $m_{\mathcal{L}}$ and have Lipschitz continuous gradients with constant $M_{\mathcal{L}}$. The strong convexity also guarantees that there is a unique optimal solution $\boldsymbol{\theta}_t^*$ at every time t, which is bounded by the following standard assumption, together with the bounded gradient norm to ensure that the losses are bounded.

Assumption 7. *The optimal solution* $\boldsymbol{\theta}^*$ *of* (4.10) *are bounded. That is* $\|\boldsymbol{\theta}^*\|_2 \leq C_\theta$. *Also, the gradient norm is upper bounded by a positive constant* G, *or equivalently,* $\|\partial \mathcal{L}_{i,t}(\boldsymbol{\theta})\|_2 \leq G, \forall i$.

We further assume that the variation of the decentralized kernel learning problem is sufficiently slow by the following assumptions to guarantee a bounded tracking error.

Assumption 8. *The time varying optimal solutions* $\boldsymbol{\Theta}_t^* := [\boldsymbol{\theta}_t^{*\top}; \ldots; \boldsymbol{\theta}_t^{*\top}] \in \mathbb{R}^{N \times 2L}$, *which is the stack of* N *copies of* $\boldsymbol{\theta}_t^{*\top}$, *and their corresponding gradients* $\nabla \mathcal{L}_t(\boldsymbol{\Theta}_t^*)$ *both have bounded variations, that is,* $\|\boldsymbol{\Theta}_t^* - \boldsymbol{\Theta}_{t-1}^*\|_F \leq \epsilon_1$ *and* $\|\nabla \mathcal{L}_t(\boldsymbol{\Theta}_t^*) - \nabla \mathcal{L}_{t-1}(\boldsymbol{\Theta}_{t-1}^*)\|_F \leq \epsilon_2$ *with finite positive constants* ϵ_1 *and* ϵ_2 *for all times* t.

For analysis, we also concatenate all local copies, state variables, and dual variables into matrices as $\boldsymbol{\Theta}_t = [\boldsymbol{\theta}_{1,t}^\top; \ldots; \boldsymbol{\theta}_{N,t}^\top] \in \mathbb{R}^{N \times 2L}$, $\hat{\boldsymbol{\Theta}}_t = [\hat{\boldsymbol{\theta}}_{1,t}^\top; \ldots; \hat{\boldsymbol{\theta}}_{N,t}^\top] \in \mathbb{R}^{N \times 2L}$, and $\boldsymbol{\Gamma}_t = [\boldsymbol{\gamma}_{1,t}^\top; \ldots; \boldsymbol{\gamma}_{N,t}^\top] \in \mathbb{R}^{N \times 2L}$, respectively. Then the updates (4.16) and (4.17) can be rewritten in matrix form as

$$\boldsymbol{\Theta}_{t+1} := \arg\min_{\boldsymbol{\Theta}} \Big\{ \mathcal{L}_t(\boldsymbol{\Theta}; \{\mathcal{S}_{i,t}\}_{i=1}^N) + \langle \boldsymbol{\Theta}, \rho \mathbf{D} \boldsymbol{\Theta} \rangle + \langle \boldsymbol{\Theta}, \boldsymbol{\Gamma}_t - \rho(\mathbf{D} + \mathbf{W}) \hat{\boldsymbol{\Theta}}_t \rangle \Big\}, \tag{4.18}$$

$$\boldsymbol{\Gamma}_{t+1} = \boldsymbol{\Gamma}_t + \rho(\mathbf{D} - \mathbf{W}) \hat{\boldsymbol{\Theta}}_{t+1}, \tag{4.19}$$

where $\mathbf{D} \in \mathbb{R}^{N \times N}$ is the diagonal matrix associated with the communication graph whose diagonal elements are $d_{ii} = |\mathcal{N}_i|$ and 0 elsewhere. $\mathbf{W} \in \mathbb{R}^{N \times N}$ is the symmetric adjacency

matrix associated with the communication graph whose (i,j)th entry is 1 when i and j are neighbors or 0 otherwise.

We first introduce the following lemma to quantify the upper bound of the asymptotic tracking error obtained by Algorithm 5. Then, we establish the regret bound for QC-DDKL. Note that Lemma 6 is adapted from [114] whose focus is on dynamic tracking. Since data arrives at each node in an online fashion, we thus view it as a dynamic learning problem. However, we should emphasize the importance of RF mapping. It is only because of the RF mapping that we can view the fixed-length data-independent parameter $\boldsymbol{\theta}$ as the model parameter to be optimized in the decentralized dynamic tracking problem [114]. Without RF mapping, the connections of our kernel learning problem with the dynamic optimization problem in [114] may not be easily established.

Lemma 6. *Suppose that assumptions 6 - 8 are satisfied, and the dual variable* $\boldsymbol{\Gamma}_1 := [\boldsymbol{\gamma}_{1,1}^\top, \ldots, \boldsymbol{\gamma}_{N,1}^\top]$ *is initialized in the column space of* $\mathbf{S}_-\mathbf{S}_-^\top$. *For the QC-DDKL updates* (4.18) *and* (4.19)*, let the penalty parameter* ρ *satisfies*

$$0 < \rho < \min\left\{ \frac{4m_{\mathcal{L}}}{\eta_1}, \frac{(\nu-1)\tilde{\sigma}_{\min}^2(\mathbf{S}_-)}{\nu\eta_3\sigma_{\max}^2(\mathbf{S}_+)}, \left(\frac{\eta_1}{4} + \frac{\eta_2\sigma_{\max}^2(\mathbf{S}_+)}{8}\right)^{-1}\left(m_{\mathcal{L}} - \frac{\eta_3\nu M_{\mathcal{L}}^2}{\tilde{\sigma}_{\min}^2(\mathbf{S}_-)}\right)\right\}, \tag{4.20}$$

where $\eta_1 > 0$, $\eta_2 > 0$, $\eta_3 > 0$, *and* $\nu > 1$ *are arbitrary constants,* $m_{\mathcal{L}}$ *and* $M_{\mathcal{L}}$ *are the minimum strong convexity constant of the local cost functions and the maximum Lipschitz constant of the local gradients, respectively.* $\sigma_{\max}(\mathbf{S}_+)$ *and* $\tilde{\sigma}_{\min}^2(\mathbf{S}_-)$ *are the maximum singular value of the unsigned incidence matrix* $\mathbf{S}_+$ *and the minimum non-zero singular value of the signed incidence matrix* $\mathbf{S}_-$ *of the network, respectively. Then, we have*

$$\begin{aligned} &\|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}_t^*\|_F \\ &\leq \left(\sqrt{m_{\mathcal{L}} - \frac{\rho\eta_1}{4}}\right)^{-1} \times \left[\left(\max\left\{\sqrt{\frac{\eta_2}{2}}, \sqrt{\frac{\rho\eta_3}{2}}\right\} + 1\right) \times \frac{g + \psi\zeta}{1 - \sqrt{1+\delta}^{-1}} + g + \sqrt{s}\zeta\right]. \end{aligned} \tag{4.21}$$

Here

$$g := \frac{\sqrt{\rho}\epsilon_1}{2}\sigma_{\max}(\mathbf{S}_+) + \frac{\epsilon_2}{\sqrt{\rho}\tilde{\sigma}_{\min}(\mathbf{S}_-)}, \tag{4.22a}$$

$$\psi := \sqrt{(\frac{\eta_3}{2} + \frac{\delta}{\rho})\frac{\rho^2}{\tilde{\sigma}^2_{\min}(\mathbf{S}_-)}\Big(\sigma^4_{\max}(\mathbf{S}_+) + \sigma^4_{\max}(\mathbf{S}_-)\Big) + s}, \tag{4.22b}$$

$$s := \frac{\rho\sigma^4_{\max}(\mathbf{S}_-)}{4\eta_1} + \frac{\rho\sigma^2_{\max}(\mathbf{S}_+)}{2\eta_2} + \frac{\sigma^2_{\max}(\mathbf{S}_-)}{2\eta_3}, \tag{4.22c}$$

$$\zeta := \max\{\sqrt{N}\varrho, \sqrt{2NL}\frac{\Delta}{2}\}, \tag{4.22d}$$

$$\delta \leq \min\left\{\frac{(\nu-1)\tilde{\sigma}^2_{\min}(\mathbf{S}_-)}{2\nu\sigma^2_{\max}(\mathbf{S}_+)} - \frac{\rho\eta_3}{2}, \Big(\frac{\rho\sigma^2_{\max}(\mathbf{S}_+)}{4} + \frac{2\nu M^2_{\mathcal{L}}}{\tilde{\sigma}^2_{\min}(\mathbf{S}_-)}\Big)^{-1}\right.$$
$$\left.\times\Big(m_{\mathcal{L}} - \frac{\rho\eta_1}{4} - \frac{\rho\eta_2\sigma^2_{\max}(\mathbf{S}_+)}{8} - \frac{\eta_3\nu M^2_{\mathcal{L}}}{\tilde{\sigma}^2_{\min}(\mathbf{S}_-)}\Big)\right\}. \tag{4.22e}$$

Lemma 6 shows the convergence of the asymptotic tracking error of $\boldsymbol{\Theta}_t$, which is influenced by the network topology (characterized by $\sigma_{\max}(\mathbf{S}_+)$ and $\tilde{\sigma}_{\min}(\mathbf{S}_-)$), slopes of the objective functions (characterized by the Lipschitz continuous gradient constant $M_{\mathcal{L}}$ and the strong convexity constant $m_{\mathcal{L}}$), the variation of the objective functions (characterized by ϵ_1 and ϵ_1), as well as the quantization and communication-censoring strategies implemented to improve the communication efficiency (characterized by ϱ and Δ, respectively). For detailed proof of the tracking error of $\boldsymbol{\Theta}_t$, interested readers are referred to [114].

Denote $\hat{\boldsymbol{f}}_{\boldsymbol{\Theta}_t}(\mathbf{x}) = [\hat{f}_{1,t}(\mathbf{x}), \ldots, \hat{f}_{N,t}(\mathbf{x})]^\top$ and $\hat{\boldsymbol{f}}_{\boldsymbol{\Theta}^*_t}(\mathbf{x}) = [\hat{f}^*_t(\mathbf{x}), \ldots, \hat{f}^*_t(\mathbf{x})]^\top$. Accordingly, we have $\hat{\boldsymbol{f}}_{\boldsymbol{\Theta}_t}(\mathbf{x}) = \boldsymbol{\Theta}_t\boldsymbol{\phi}_L(\mathbf{x})$ and $\hat{\boldsymbol{f}}_{\boldsymbol{\Theta}^*_t}(\mathbf{x}) = \boldsymbol{\Theta}^*_t\boldsymbol{\phi}_L(\mathbf{x})$, respectively. Then, we have

$$\begin{aligned}\|\hat{\boldsymbol{f}}_{\boldsymbol{\Theta}_t}(\mathbf{x}) - \hat{\boldsymbol{f}}_{\boldsymbol{\Theta}^*_t}(\mathbf{x})\|_2 &= \|\boldsymbol{\Theta}_t\boldsymbol{\phi}_L(\mathbf{x}) - \boldsymbol{\Theta}^*_t\boldsymbol{\phi}_L(\mathbf{x})\|_2 \\ &\leq \|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}^*_t\|_2\|\boldsymbol{\phi}_L(\mathbf{x})\|_2 \\ &\leq \|\boldsymbol{\Theta}_t - \boldsymbol{\Theta}^*_t\|_2,\end{aligned} \tag{4.23}$$

where the second inequality comes from the fact that $\|\boldsymbol{\phi}_L(\mathbf{x})\|_2 \leq 1$ with the adopted RF mapping. Therefore, with Lemma 6, we can directly conclude that QC-DDKL is able to asymptoticly track the error of the the nonlinear functions $\hat{f}_{i,t}, \forall i, t$..

Remark 1. Note that there exists an approximately linear dependence of the asymptotic tracking error on $\zeta := \max\{\sqrt{N}\varrho, \sqrt{2NL}\frac{\Delta}{2}\}$, which is determined by the communication-censoring error and the quantization error, see Chapter 3.4 for the derivation of this bound and proof.

With Lemma 6, we then establish the dynamic regret of our QC-DDKL algorithm in the following theorem.

Theorem 10. *Under the assumptions 6 - 8, and setting $\rho = \mathcal{O}(1/T)$, then the dynamic regret* (4.8) *generated by the updates* (4.16) *and* (4.17) *satisfies*

$$\mathbf{Reg}_T^D \leq \alpha G \mathcal{O}(\sqrt{T}), \tag{4.24}$$

where α is an implicit parameter that incorporates the effects of communication censoring and quantization.

Proof. The proof is relatively straightforward. For notational brevity, we denote $\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}; \mathcal{S}_{i,t})$ and $\mathcal{L}_{i,t}(\boldsymbol{\theta}_t^*; \mathcal{S}_{i,t})$ by $\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t})$ and $\mathcal{L}_{i,t}(\boldsymbol{\theta}_t^*)$, respectively. The dynamic regret (4.8) is then rewritten as

$$\begin{aligned}
\mathbf{Reg}_T^D &= \sum_{i=1}^{N}\sum_{t=1}^{T}(\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}; \mathcal{S}_{i,t}) - \mathcal{L}_{i,t}(\boldsymbol{\theta}_t^*; \mathcal{S}_{i,t})) \\
&= \sum_{t=1}^{T}\left(\sum_{i=1}^{N}\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t}) - \sum_{i=1}^{N}\mathcal{L}_{i,t}(\boldsymbol{\theta}_t^*)\right) \\
&= \sum_{t=1}^{T}\left(\mathcal{L}_t(\boldsymbol{\Theta}_t) - \mathcal{L}_t(\boldsymbol{\Theta}_t^*)\right),
\end{aligned} \tag{4.25}$$

where $\mathcal{L}_t(\boldsymbol{\Theta}_t) := \sum_{i=1}^{N}\mathcal{L}_{i,t}(\boldsymbol{\theta}_{i,t})$ and $\mathcal{L}_t(\boldsymbol{\Theta}_t^*) := \sum_{i=1}^{N}\mathcal{L}_{i,t}(\boldsymbol{\theta}_t^*)$.

Then, by Assumption 6 and Assumption 7, we have

$$\mathcal{L}_t(\mathbf{\Theta}_t) - \mathcal{L}_t(\mathbf{\Theta}_t^*) \leq G\|\mathbf{\Theta}_t - \mathbf{\Theta}_t^*\|_F. \tag{4.26}$$

Summing up both sides of (4.26) from 1 to T, we have the dynamic regret, which is bounded by

$$\begin{aligned} \mathbf{Reg}_T^D &= \sum_{t=1}^{T}(\mathcal{L}_t(\mathbf{\Theta}_t) - \mathcal{L}_t(\mathbf{\Theta}_t^*)) \\ &\leq G\sum_{t=1}^{T}\|\mathbf{\Theta}_t - \mathbf{\Theta}_t^*\|_F. \end{aligned} \tag{4.27}$$

Setting $\rho = \mathcal{O}(1/T)$, we can further upper bound (4.21) by

$$\|\mathbf{\Theta}_t - \mathbf{\Theta}_t^*\|_F \leq \alpha\mathcal{O}(\frac{1}{\sqrt{T}}), \tag{4.28}$$

where α incorporates the effects of communication censoring and quantization.

Thus the sublinear dynamic regret is achieved:

$$\begin{aligned} \mathbf{Reg}_T^D &\leq G\sum_{t=1}^{T}\|\mathbf{\Theta}_t - \mathbf{\Theta}_t^*\|_F \\ &\leq GT\alpha\mathcal{O}(\frac{1}{\sqrt{T}}) \\ &\leq \alpha G\mathcal{O}(\sqrt{T}), \end{aligned} \tag{4.29}$$

which completes the proof. ■

4.5 Simulation Results

This section evaluates the performance of our proposed QC-DDKL algorithm in regression tasks for streaming data from real-world datasets.

Datasets. We carry out regression tasks on two datasets available at the UCI machine learning repository [86]: (i) Tom's hardware ($T_{total} = 9725$, $d = 96$); (ii) Twitter ($T_{total} = 98700$, $d = 77$). The detailed description on both datasets is available in Appendix 3.7.2.

Benchmarks. Although RFFDOKL proposed by [58] and the DOKL-ADMM algorithm developed by [109] are not specified for dynamic kernel learning, we simulate them in this experiment to show the advantages of DDKL-based algorithm in dynamic kernel learning problems. Details about those benchmarks are listed below.

- **RFFDOKL:** A gradient descent based online kernel learning algorithm proposed by [58].
- **DOKL-ADMM:** An ADMM based online kernel learning algorithm proposed by [109].
- **DDKL:** The vanilla decentralized dynamic kernel learning algorithm described by (4.11) and (4.12),
- **CoDDKL:** The communication-censored DDKL algorithm that applies communication-censoring at each agent to evaluate the informativeness of its local primal update.
- **QDDKL:** The quantized DDKL algorithm that applies difference quantization at each agent to save communication cost.

Settings and parameter tuning. All experiments are conducted using Matlab 2021 on an Intel CPU @ 3.6 GHz (32 GB RAM) desktop. For each dataset, the T_{total} data samples are randomly shuffled and then partitioned among N nodes so that each node has $T = T_{total}/N$ samples. The features are normalized so that all values are between 0 and 1. The number of random features adopted for RF approximation is $L = 50$ throughout the simulations. The Gaussian kernel bandwidth is fined tuned to be $\sigma = 0.5$ for both datasets.

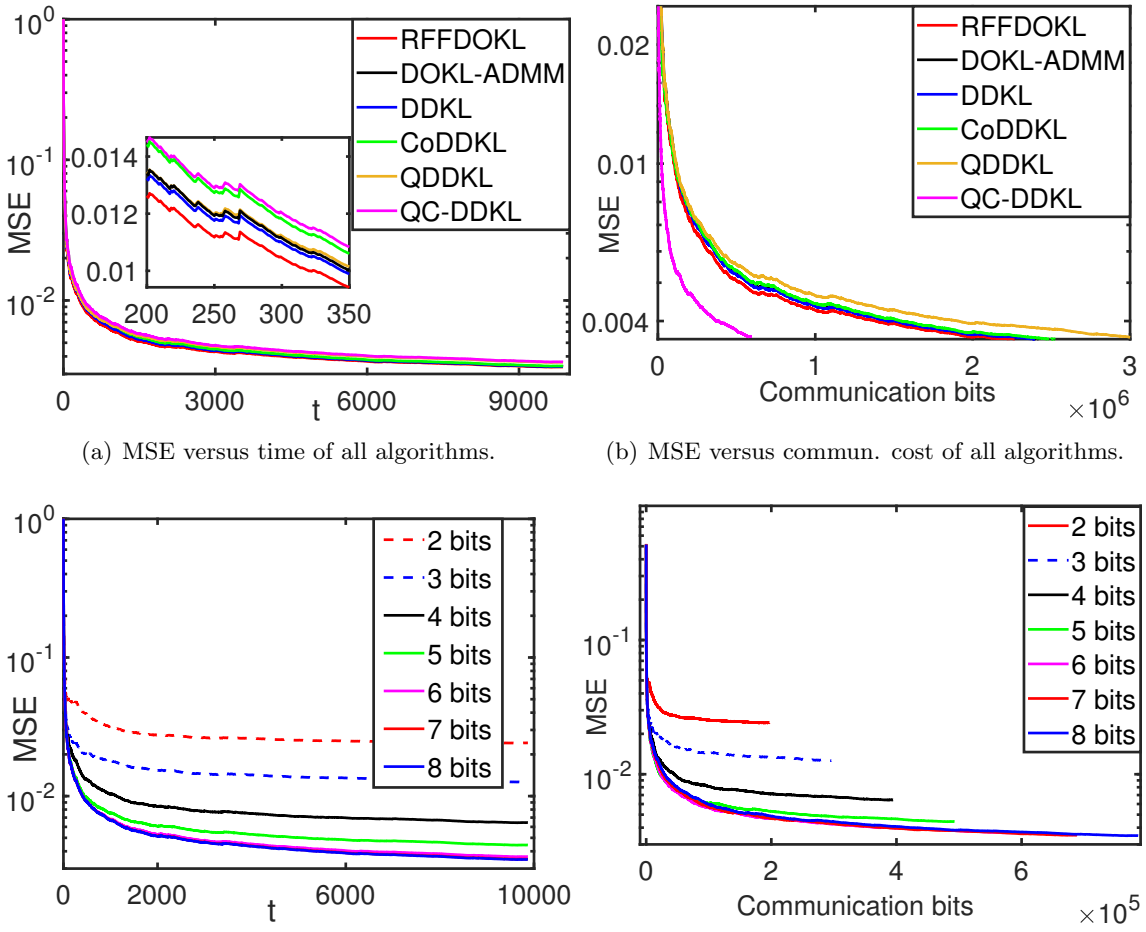
The regularization parameter $\lambda = 10^{-3}$. Using evaluation data, the stepsize ρ and η_t are fine-tuned via grid-search for each method and each dataset individually. The connected graph is randomly generated with $N = 5$ nodes for Tom's hardware dataset and $N = 10$ nodes for Twitter dataset. The censoring threshold parameters are $\varrho = 0.1, \mu = 0.9$ for Twitter data, and $\varrho = 0.1, \mu = 0.95$ for Tom's hardware data.

The learning performance of Twitter dataset and Tom's hardware dataset are shown in Figure 4.1 and Figure 4.2, respectively. For Twitter dataset, the MSE performance versus time of DDKL (Figure 4.1(a)) is comparable to that of RFFDOKL and DOKL-ADMM, although not better. This may because the environment from which this dataset is collected is static. While for Tom's hardware dataset, the MSE performance versus time of DDKL (Figure 4.2(a)) is better than that of RFFDOKL and DOKL-ADMM. This also makes sense. Although both datasets are buzz prediction tasks, Twitter dataset may have less dynamics than Tom's hardware because Twitter is a social platform with much more users than the forum platform Tom's hardware. The communication efficiency of QC-DDKL in terms of total bits communicated is better than that of all other benchmarks'. For Figure 4.1 (a) and (b), we use a 6-bit quantization level. This quantization level is chosen based on Figure 4.1(c) and (d), from which we can see that the 6-bit quantization level achieves comparable MSE performance as 7-bit or 8-bit quantization while its MSE performance versus total communication cost is the best. For Figure 4.2 (a) and (b), we use a 5-bit quantization level, with the same rationale as Twitter dataset.

In summary, the simulation results corroborate that our proposed DDKL-based algorithm can capture the dynamics in online streaming learning tasks while our QC-DDKL algorithm can achieve the desired communication efficiency.

4.6 Concluding Remarks

This chapter studies the decentralized dynamic kernel learning problem under communication constraints for multi-agent systems. We utilize the random feature mapping to



(a) MSE versus time of all algorithms.

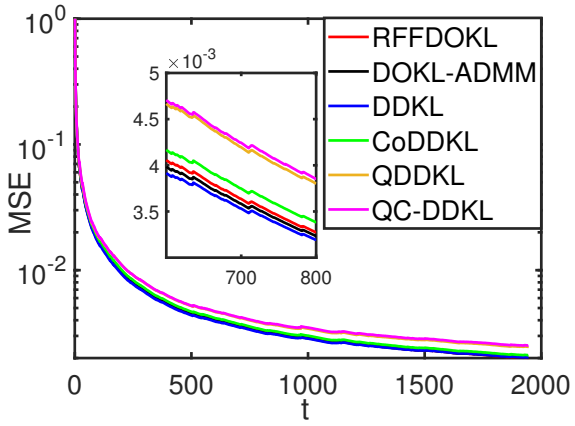
(b) MSE versus commun. cost of all algorithms.

(c) MSE performance of QCDDKL with varying quantization level.

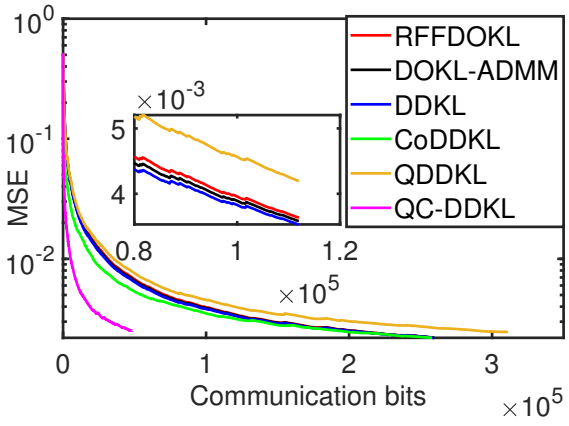
(d) Commun. cost of QCDDKL with varying quantization level.

Figure 4.1: Learning performance on Twitter dataset.

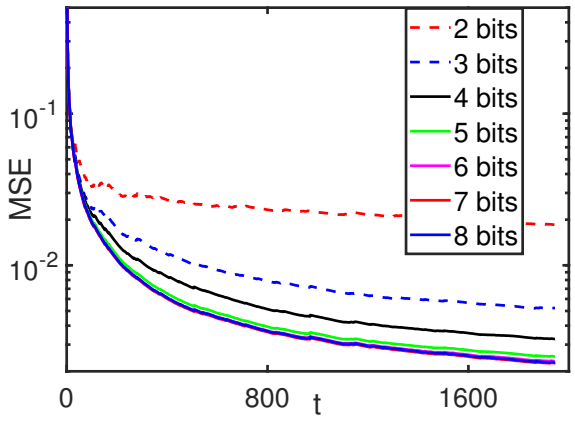
formulate the decentralized dynamic kernel learning problem as a decentralized dynamic convex optimization problem. We then adopt the alternating direction method of multipliers, a quantization strategy, and a communication censoring strategy to develop a communication-efficient decentralized dynamic kernel learning algorithm. We prove that by properly choosing the learning rate, the developed algorithm achieves a sublinear dynamic regret bound. The learning performance and communication efficiency are demonstrated via simulations on various real datasets.



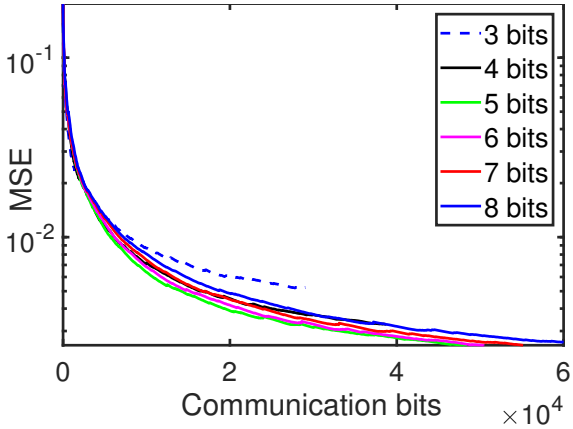
(a) MSE versus time of all algorithms.



(b) MSE versus commun. cost of all algorithms.



(c) MSE performance of QCDDKL with varying quantization level.



(d) Commun. cost of QCDDKL with varying quantization level.

Figure 4.2: Learning performance on Tom dataset.

Chapter 5: Deep Kernel Learning Networks: A Unified Framework of Learning Nonlinear Input-Output Maps

5.1 Introduction

Kernel methods and neural networks (NNs) are both attractive in various learning tasks such as regression, classification, clustering, as well as reinforcement learning [13, 14]. The success owes to their abilities to model the (complex) nonlinear relationships behind the input-output samples $\{\mathbf{x}_t, y_t\}_{t=1}^T$, with $\mathbf{x}_t \in \mathbb{R}^d$ and $y_t \in \mathbb{R}$. That is, they are able to find a function f, which is often nonlinear, by minimizing the total cost $\hat{R} := \sum_{t=1}^T \ell(f(\mathbf{x}_t), y_t)$. Here $\ell(f(\mathbf{x}_t), y_t)$ is a non-negative cost function that measures the discrepancy between the predicted value $f(\mathbf{x}_t)$ and the true value y_t.

The difference between kernel methods and NNs is how f is represented, which classifies them into two categories, i.e., the non-parametric and parametric methods. As one of the representatives of non-parametric methods, kernel methods are able to model a highly nonlinear function $f(\cdot)$ that belongs to a reproducing kernel Hilbert space(RKHS) $\mathcal{H}$. According to the Representer theorem, the optimal solution that minimizes the total cost $\hat{R}$ has the form $\hat{f}_\kappa(\mathbf{x}) = \sum_{t=1}^T \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t)$, which is linear in the model parameters $\{\alpha_t\}$ [77]. On the other hand, an NN of N layers approximates $f(\cdot)$ as a composite of functions of all layers: $\hat{f}_{\text{NN}}(\mathbf{x}; \mathbf{W}) = h_N(W_N h_{N-1}(W_{N-1} \cdots h_1(W_1 \mathbf{a}^{[0]}) \cdots))$, where $\mathbf{a}^{[0]} = \mathbf{x}$, W_n and h_n are the linear and nonlinear operators of the n-th layer, respectively, and $\mathbf{W}$ denotes the ensemble of parameters of all layers [132].

While both NNs and kernel methods are appealing in a wide range of areas, their strengths and weaknesses are different. Learning algorithms that utilize linearly-parameterized kernel methods often arise from convex optimization formulations, and hence are amenable

to theoretical analysis, permit globally optimal parameters, and enjoy rigorous statistical learning guarantees [77]. However, the curse of dimensionality issue prevents their application in large-scale learning scenarios. Moreover, the choice of pre-defined kernel κ also affects the learning performance of kernel-based learning algorithms. In practice, it is not always feasible that the selected kernel is suitable for a given task, especially when data are taken from complex and possibly unknown distributions. On the other hand, the representation power of NNs can be adjusted by the depth and width of their architecture, as well as the activation functions. However, optimizing NN models involves solving nonconvex functions; as a result, they rely much on intuition, heuristics, and trial-and-error, and our theoretical understanding of them is still incomplete [133]. Though both methods are broadly investigated, there is a lack of explainability on why one outperforms the other in different tasks and which one should be chosen for various tasks. Some preliminary efforts in seeking the connection between kernel methods and infinite NNs have been found in [134,135], but such an infinite network size leads to impractical setting and implementation.

Motivated by these observations, in this chapter, we adopt the random feature (RF) mapping method to circumvent the curse of dimensionality issue in kernel methods [50], which also enables us to implement the kernel learning with a finite number of neurons as a two-layer NN (termed RF-KL), at drastically reduced workload on weight training. Different from existing work where the representation abilities of RF-KL and NNs are explored and compared for high degree polynomials [136, 137], this chapter reviews them from a different angle. We show that RF-KL can be viewed as a special case of NNs with much improved computational simplicity. We further develop a unified framework with a low-computation advantage, which leads to the desired trade-off between the capability of representation power and the control of training overload. We then extend RF-KL to deep kernel learning (DKL) by performing RF mapping at each layer and training the last output layer only. To increase the representation power of DKL, we add multiple trainable paths to connect all hidden layers with the output layer. It leads to a deep kernel learning network structure with multiple learning paths (DKL-MLP), whose output functional is linear in

the trainable multi-path model parameters. In this way, DKL-MLP benefits not only from the depth of DKL but also from the (implicit) flexibility and computational advantage of RF-based multi-kernel learning.

5.1.1 Related work

To put our work in context, we review prior art under four categories.

Random feature based kernel learning. To mitigate the computational complexity of kernel methods, various techniques are developed, including stochastic approximation [36, 47], restricting the number of function parameters [38], and approximating the kernel during training [49–52]. Among them, random feature (RF) mapping methods have gained popularity thanks to their ability to map the large-scale data into an RF space of much reduced dimension by approximating the kernel with a fixed (small) number of random features, which thus circumvents the curse of dimensionality problem [50–52]. Enforcing orthogonality on random features can greatly reduce the error in kernel approximation [53,54]. The learning performance of RF-based methods is evaluated in [55, 56, 138].

Deep kernel learning in kernel space. Deep kernel learning combines the non-parametric flexibility of kernel methods and the multi-layer nature of DNNs and has found successful applications in many tasks, such as image annotation problem [139], classification [140,141], and regression [142]. [141] proposes a deep kernel method that can learn the kernel function from data using DNN, which appears to be an application of DNN on kernel learning. [140, 142] utilize the Gaussian process as the base kernel and take features produced by a DNN to train the resulting model end-to-end. Most of the literature on deep kernel learning in original kernel space goes no further than adopting the depth architecture of DNN but does not tackle the curse of dimensionality issue inherent in non-parametric learning. Further, they do not explore the theoretical properties such as universality and generalization properties of functions learned by the deep kernel networks.

Deep kernel learning via random features. To better tackle the learning tasks and

circumvent the curse of dimensionality issue in traditional kernel methods, a deep structure is usually incorporated with the RF-based kernel methods [143–145]. In [143], random features are adopted to address the scalability issue in phone recognition and speech understanding tasks by stacking the kernel modules to form a deep architecture. [144] proposes a deep hybrid NN structure where a trainable layer and a fixed random layer are concatenated. In [145], the random Fourier feature layer is proposed as a building block of the deep architecture. Although initialized from a Gaussian distribution, their random features are trained end-to-end through back-propagation for the goal of kernel learning, thus the final optimized RF-related parameters may not follow Gaussian distribution anymore. A deep semi-random network is proposed in [146], where each layer consists of both trainable and fixed parameters, which differs from [144]. More recently, generative models and their extension to a multi-layer structure are employed to jointly solve the learning task and learn the random Fourier features [147]. Note that among all recent works, the deep semi-random kernel method shares some similarities with our proposed method. However, their method requires much more computational resources for their trainable weights to get updated compared with our method, due to back-propagation through all layers.

5.1.2 Contributions

This chapter proposes a unified framework for nonlinear input-out put maps as well as a deep kernel learning network with multiple learning paths. Relative to the prior art, our contributions are summarized as follows.

- We leverage the RF mapping method to implement kernel learning as a two-layer neural network (termed RF-KL), which enables the development of a unified framework with low-computation advantage, with a desired trade-off between the capability of representation power and the control of training overload.
- The RF-KL method is extended to a deep kernel learning (DKL) method by performing RF mapping at each hidden layer and training the last output layer only, which

can be seen as a randomized DNN with only the last layer trainable (DKL). Compared with extreme learning machine [148], the randomized parameters are generated from specific distributions related to pre-selected kernels. Statistical accuracy guarantees of standard kernel methods are applicable to the RF-based kernel methods.

- We add multiple trainable paths to DKL to increase its expressiveness and develop the DKL-MLP method. These paths directly connect the hidden layer with the output layer in a linear manner and do not change the convexity of the learning problem if the original DKL is convex. Moreover, updating all trainable parameters in DKL-MLP does not involve back-propagation, the gradient diminishing problem is thus avoided and the computational complexity is much reduced compared with [146].

- We provide theoretical analysis in terms of universality to show that the developed algorithms can represent a wide variety of interesting functions with arbitrarily small errors and have no bad local minimum, which is lacking in most of the current deep kernel learning work. In addition, we test the performance of our proposed algorithm on real datasets to solve both the classification and regression tasks. The results corroborate that DKL-MLP enjoys both good generalization performance and low computational complexity.

5.2 Problem Statement

Consider the learning task that aims to find a nonlinear function $f \in \Omega$ such that $y_t = f(\mathbf{x}_t) + e_t$ for the ensemble of data samples $\{\mathbf{x}_t, y_t\}_{t=1}^T$, with $\mathbf{x}_t \in \mathbb{R}^d$ and $y_t \in \mathbb{R}$, and the error term e_t are minimized accordingly to certain optimality metric. The optimal function f is obtained by solving the following optimization problem:

$$\min_{f \in \Omega} \quad \hat{R}(f) := \frac{1}{T} \sum_{t=1}^{T} \ell(f(\mathbf{x}_t), y_t) + \lambda \|f\|_{\Omega}^2, \tag{5.1}$$

where $\ell(\cdot,\cdot)$ is a loss function, $\|\cdot\|_\Omega$ is the norm associated with the function space Ω, and $\lambda > 0$ is a regularization parameter that controls over-fitting. Depending on different tasks, $\ell(\cdot,\cdot)$ can be selected to be the least-squares in regression tasks or the logistic or the hinge loss in classification tasks.

5.3 A Unified Learning Framework

This section first reviews the standard NN and kernel methods. Then, the random feature based kernel method is implemented as a two-layer NN, which leads to a unified learning framework.

5.3.1 Learning with a standard NN

To minimize the total cost in (5.1), a NN of M layers approximates the nonlinear function f as a composite of functions learned from a cascade of all layers, where each layer consists of a linear and a nonlinear operator. Take a fully connected two-layer neural network for illustration, whose structure is given in Figure 5.1. The approximated function $\hat{f}_{\text{NN}}$ admits

$$\hat{f}_{\text{NN}}(\mathbf{x};\mathbf{W}) = h_2(\mathbf{W}^{[2]}h_1(\mathbf{W}^{[1]}\mathbf{x}+\mathbf{b}^{[1]})+\mathbf{b}^{[2]}). \tag{5.2}$$

Here $\{\mathbf{W}^{[m]},\mathbf{b}^{[m]}\}$ and h_m are the linear and nonlinear operator of the m-th layer, respectively. Depending on different applications, h_m can be selected to be, e.g., Relu, Sigoid or Tanh functions, and $\mathbf{W} = \{\mathbf{W}^{[1]},\mathbf{W}^{[2]},\mathbf{b}^{[1]}\mathbf{b}^{[2]}\}$ denotes the ensemble of weight matrices and bias.

The total cost in (5.1) can then be reformulated as

$$\hat{R}(\mathbf{W}) := \frac{1}{T}\sum_{t=1}^{T}\ell(\hat{f}_{\text{NN}}(\mathbf{x}_t;\mathbf{W}),y_t)+\lambda\|\mathbf{W}\|_2^2. \tag{5.3}$$

The network parameters are then updated using gradient descent through backpropagation.

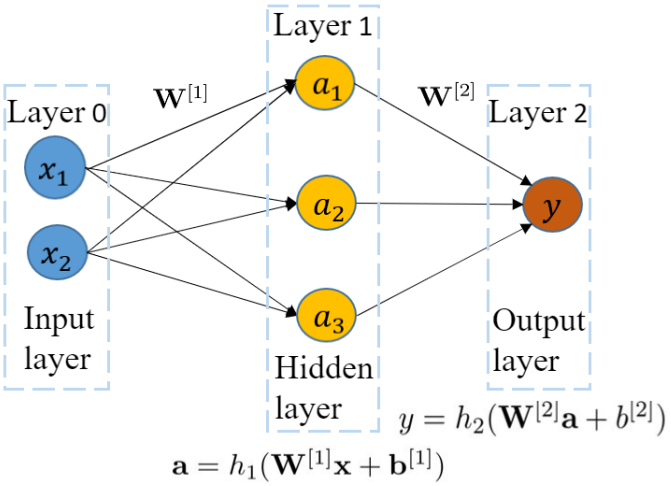


Figure 5.1: Structure of a two-layer neural network.

5.3.2 Learning with kernel methods

Alternatively, kernel methods can be utilized to approximate f given the assumption that $f \in \mathcal{H}$. Then, by the Representer theorem [77], the optimal solution of (5.1) admits

$$\hat{f}_\kappa(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t) := \boldsymbol{\alpha}^\top \boldsymbol{\kappa}(\mathbf{x}), \tag{5.4}$$

where $\boldsymbol{\kappa}(\mathbf{x}) = [\kappa(\mathbf{x}, \mathbf{x}_1), \ldots, \kappa(\mathbf{x}, \mathbf{x}_T)]^\top$ with $\kappa(\mathbf{x}, \mathbf{x}_t)$ measures the similarity between $\mathbf{x}$ and $\mathbf{x}_t$, and $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_T]^\top \in \mathbb{R}^T$ is the coefficient vector to be learned.

In this way, the total cost in (5.1) can be reformulated as a function of $\boldsymbol{\alpha}$:

$$\hat{R}(\boldsymbol{\alpha}) := \frac{1}{T} \sum_{t=1}^{T} \ell(\boldsymbol{\alpha}^\top \boldsymbol{\kappa}(\mathbf{x}_t), y_t) + \lambda \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}, \tag{5.5}$$

where $\mathbf{K} \in \mathbb{R}^{T \times T}$ has entry $[\mathbf{K}]_{t,t'} = \kappa(\mathbf{x}_t, \mathbf{x}_{t'})$.

5.3.3 RF-KL implemented using a two-layer NN

The connection of kernel methods and NNs with infinite number of neurons has been found in [134, 135]. However, the direct implementation of kernel methods with a NN is infeasible in practice due to the infinite network size. To overcome this problem, we adopt the RF mapping method which enables us to build a unified learning framework of applying kernel methods over a NN. Specifically, it maps the kernel function $\kappa(\mathbf{x}, \mathbf{x}_t)$ in (5.4) by the sample average

$$\hat{\kappa}_L(\mathbf{x}, \mathbf{x}_t) := \frac{1}{L} \sum_{l=1}^{L} \phi(\mathbf{x}, \boldsymbol{\omega}_l) \phi(\mathbf{x}_t, \boldsymbol{\omega}_l), \tag{5.6}$$

where $\phi(\mathbf{x}, \boldsymbol{\omega}) = [\cos(\boldsymbol{\omega}^\top \mathbf{x}), \sin(\boldsymbol{\omega}^\top \mathbf{x})]^\top$ or $\phi(\mathbf{x}, \boldsymbol{\omega}) = \sqrt{2}\cos(\boldsymbol{\omega}^\top \mathbf{x} + b)$. Here $\{\boldsymbol{\omega}_l\}_{l=1}^L$ are randomly drawn from $p_\kappa(\boldsymbol{\omega})$, which is the inverse Fourier transform of selected kernel κ, and b is drawn uniformly from $[0, 2\pi]$. For a Gaussian kernel $\kappa(\mathbf{x}_t, \mathbf{x}_{t'}) = \exp(\|\mathbf{x}_\mathrm{t} - \mathbf{x}_{\mathrm{t}'}\|_2^2/(2\sigma^2))$, we have $p_\kappa(\omega) \sim \mathcal{N}(\mathbf{0}, \sigma^{-2}\mathbf{I})$.

Then, the function $\hat{f}_\kappa$ in (5.4) can be expressed as

$$\hat{f}_{\mathrm{RF}}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \boldsymbol{\phi}_L^\top(\mathbf{x}_t) \boldsymbol{\phi}_L(\mathbf{x}) = \boldsymbol{\theta}^\top \boldsymbol{\phi}_L(\mathbf{x}), \tag{5.7}$$

where $\boldsymbol{\phi}_L(\mathbf{x}) = \sqrt{\frac{1}{L}}[\phi(\mathbf{x}, \boldsymbol{\omega}_1), \ldots, \phi(\mathbf{x}, \boldsymbol{\omega}_L)]^\top$ and $\boldsymbol{\theta} := \sum_{t=1}^{T} \alpha_t \boldsymbol{\phi}_L(\mathbf{x}_t)$ denotes the new decision vector to be learned in the RF space. Note that the size of $\boldsymbol{\theta}$ is fixed and does not increase with the number of data samples.

The total cost in (5.1) can then be reformulated as

$$\hat{R}(\boldsymbol{\theta}) : = \frac{1}{T} \sum_{t=1}^{T} \ell(\boldsymbol{\theta}^\top \boldsymbol{\phi}_L(\mathbf{x}_t), y_t) + \lambda \|\boldsymbol{\theta}\|_2^2. \tag{5.8}$$

The algorithm that solves (5.8) is termed as RF-KL. Comparing RF-KL and the standard

Table 5.1: Unification and comparison of RF-KL and standard NNs

	Standard two layer NNs	A NN implementation of RF-KL
input	$\mathbf{x} \in \mathbb{R}^d$	
first layer	$\mathbf{a} = h_1(\mathbf{W}^{[1]}\mathbf{x} + \mathbf{b}^{[1]})$	
second layer	$y = h_2(\mathbf{W}^{[2]}\mathbf{a} + b^{[2]})$	
activation h_1	relu, tanh, or sigmoid	cosine function
activation h_2	linear or sigmoid	
cost function	(5.3)	(5.8)
parameters	$\mathbf{W}^{[1]} \in \mathbb{R}^{m_1 \times d}$, $\mathbf{b}^{[1]} \in \mathbb{R}^{m_1}$ $\mathbf{W}^{[2]} \in \mathbb{R}^{1 \times m_1}$ and $b^{[2]} \in \mathbb{R}$ m_1: # of neurons of first layer	$\mathbf{W}^{[1]} = \{\boldsymbol{\omega}_l\}_{l=1}^L \in \mathbb{R}^{L \times d}$, $\mathbf{b}^{[1]} \in \mathbb{R}^L$ $\mathbf{W}^{[2]} = \boldsymbol{\theta}^\top \in \mathbb{R}^{1 \times L}$, $b^{[2]} = 0$ L: # of random features
initialization	all parameters are randomly initialized	$\mathbf{W}^{[1]}$ and $\mathbf{b}^{[1]}$ are prefixed $\mathbf{W}^{[2]}$ are randomly initialized
update at kth iteration	update all parameters $\mathbf{W}^{k+1} = \mathbf{W}^k - \eta \nabla \hat{R}_{\mathbf{W}^k}(\mathbf{W}^k)$	only update $\mathbf{W}^{[2]} = \boldsymbol{\theta}^\top$ $\boldsymbol{\theta}^{k+1} = \boldsymbol{\theta}^k - \eta \nabla \hat{R}_{\boldsymbol{\theta}^k}(\boldsymbol{\theta}^k)$

two-layer NN, we notice that with RF mapping, kernel methods play as a special case of a two-layer NN, where the random features $\{\boldsymbol{\omega}\}_{l=1}^L$ are equivalent to the weights $\mathbf{W}^{[1]}$ of the first layer, the parameters $\boldsymbol{\theta}$ to be learned are equivalent to the weights $\mathbf{W}^{[2]}$ of the second layer, and the activation function of first layer is cosine. Depending on the objective function (5.8), there may exist a closed-form solution for $\boldsymbol{\theta}$ [18]. Otherwise, the model parameters can be updated through gradient descent. The two approaches are summarized in Table 5.1 for comparison.

5.3.4 Unification of RF-KL and standard NNs

The above sections naturally lead to a unified framework for RF-based kernel methods and NNs, where the RF-KL is implemented with a two-layer NN whose network structure is shown in Figure 5.1.

Remark 1 (Complexity). Note that in RF-KL, the first layer's weights are fixed and do not need to be trained. If there does not exist a closed-form solution, we only need to train the weights in the second/output layer. Therefore, compared to a standard NN where weights from all layers need to be trained, RF-KL can afford to have a larger network

size given the same training overload and computation complexity. For distributed learning, RF-KL drastically saves communication costs in exchanging the training parameters among network nodes. Moreover, RF-KL does not need back-propagation to train the weights since they are in the output layer, hence bypassing the diminishing gradient issue.

Remark 2 (Representation power). The simplicity of RF-KL coming from the fixed weights and the activation function of the first layer is a double-sword. In the extreme case where only one kernel is applied in a NN with one hidden layer, RF-KL has limited representation power and results in degraded learning performance compared to the standard NN.

Remark 3 (Generalization and unification). Balancing the above two remarks, our unified framework of RF-KL and NNs enables to achieve a desired trade-off between complexity and representation power, by enlarging the network width with multiple kernels, or by expanding the network depth with multiple layers.

5.4 Deep Kernel Learning Networks with Multiple Learning Paths

With RF mapping, a vanilla deep kernel learning (DKL) network of M layers is designed and modeled by

$$\hat{f}_M(\mathbf{x}) = \boldsymbol{\vartheta}^\top \boldsymbol{\phi}^{M-1}(\cdots(\boldsymbol{\phi}^1(\mathbf{x}))), \tag{5.9}$$

where $\boldsymbol{\phi}^1(\mathbf{x}) = \sqrt{\frac{1}{L_1}}[\phi(\mathbf{x};\boldsymbol{\omega}_1^1, b_1^1), \ldots, \phi(\mathbf{x};\boldsymbol{\omega}_{L_1}^1, b_{L_1}^1)]^\top$ performs RF mapping to the input and $\boldsymbol{\phi}^2(\boldsymbol{\phi}^1(\mathbf{x}))$ performs RF mapping to $\boldsymbol{\phi}^1(\mathbf{x})$, until the last hidden layer. Each hidden layer has L_m neurons that do not have to be the same across all layers, and $\boldsymbol{\vartheta} \in \mathbb{R}^{L_{M-1}}$ are trainable parameters that connect the last hidden layer and the output layer.

Compared with a standard DNN of the same network structure, DKL enjoys drastically reduced workload on weight training since $\boldsymbol{\omega}_l^m$ and b_l^m, $\forall l, m$, are fixed once the kernel functions of all layers are fixed, and it only needs to train the last layer's parameters $\boldsymbol{\vartheta}$.

However, the simplicity coming from the fixed weights is a double-sword, and DKL may not have enough representation power for complex learning tasks.

To enhance the learning ability of DKL, we add multiple paths to connect the output of each hidden layer with the output layer and form the deep kernel learning with multiple learning paths (DKL-MLP) model. Compared with DKL, DKL-MLP has more trainable weights resulting from multiple paths. The network structure is shown in Figure 5.2 and can be modeled as

$$\tilde{f}_M(\mathbf{x}) = \hat{f}_{M-1}(\mathbf{x}) + \hat{f}_{M-2}(\mathbf{x}) + \cdots + \hat{f}_1(\mathbf{x}) = \sum_{m=1}^{M-1} \hat{f}_m(\mathbf{x}), \tag{5.10}$$

where $\hat{f}_m(\mathbf{x}) := \boldsymbol{\vartheta}^{m\top}\boldsymbol{\phi}^m(\cdots(\boldsymbol{\phi}^1(\mathbf{x}))$ with learnable parameters $\boldsymbol{\vartheta}^m \in \mathbb{R}^{L_m}$ associated with the mth hidden layer.

Remark 4 (Difference between DKL-MLP and deep residual networks). The multiple-learning-path (MLP) model differs from the deep residual network [149] in that short cuts in a deep residual network skip learning for some layers to avoid the vanishing gradient problem. In contrast, we employ $\boldsymbol{\vartheta}^1, \ldots, \boldsymbol{\vartheta}^{M-2}$ as trainable parameter vectors to enhance the learning performance.

Remark 5 (Difference between DKL-MLP and deep semi-random networks). It should be noted that our work differs from the deep semi-random network proposed in [146], which formulates a nonconvex learning problem for the parameter estimation. Moreover, updating all trainable parameters in DKL-MLP does not involve back-propagation. On the other hand, [146] requires much more computational resources for their trainable weights to get updated due to back-propagation through all layers.

5.5 Theoretical Analysis

This section analyzes the generalization performance of RF-KL for both the regression and binary classification tasks, as well as the university of DKL and DKL-MLP.

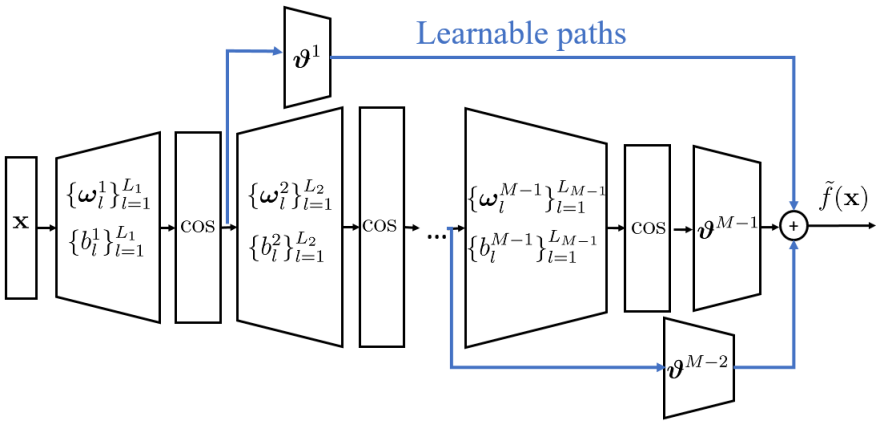


Figure 5.2: A general structure of DKL-MLP with M layers.

Theorem 11. *Let $\lambda_{\mathbf{K}}$ be the largest eigenvalue of the kernel matrix $\mathbf{K}$ in (5.5), and choose the regularization parameter $\lambda < \lambda_{\mathbf{K}}/T$ so as to control over-fitting. Assume there exists $f_{\mathcal{H}} \in \mathcal{H}$, such that for all estimators $f \in \mathcal{H}$, $\mathcal{E}(f_{\mathcal{H}}) \leq \mathcal{E}(f)$, where $\mathcal{E}(f) := \mathbb{E}_p\left[\ell(f(\mathbf{x}), y)\right]$ is the expected risk to measure the generalization ability of the estimator f. Then, for all $\delta_p \in (0,1)$ and $\|f\|_{\mathcal{H}} \leq 1$, there exists $\epsilon \in (0,1)$ such that*

- *(regression problem) if the number of random features L satisfies*

$$L \geq \frac{1}{\lambda}\left(\frac{1}{\epsilon^2} + \frac{2}{3\epsilon}\right) \log \frac{16 d_{\mathbf{K}}^{\lambda}}{\delta_p},$$

  then with probability at least $1 - \delta_p$, the excess risk of $\mathcal{E}(\hat{f}_{\mathrm{RF}})$ obtained by RF-KL can be upper bounded as

$$\mathcal{E}(\hat{f}_{\mathrm{RF}}) - \mathcal{E}(f_{\mathcal{H}}) \leq 3\lambda + \mathcal{O}(\tfrac{1}{\sqrt{T}}); \tag{5.11}$$

- *(binary classification problem) if the number of random features L satisfies*

$$L \geq \frac{5}{\lambda} \log \frac{16 d_{\mathbf{K}}^{\lambda}}{\delta_p},$$

then with probability at least $1-\delta_p$*, the excess risk of* $\mathcal{E}(\hat{f}_{\mathrm{RF}})$ *obtained by RF-KL can be upper bounded as*

$$\mathcal{E}(\hat{f}_{\mathrm{RF}}) - \mathcal{E}(f_{\mathcal{H}}) \leq \lambda + \sqrt{\lambda} + \mathcal{O}(\frac{1}{\sqrt{T}}). \tag{5.12}$$

Here $d_{\mathbf{K}}^{\lambda} := \mathrm{Tr}(\mathbf{K}(\mathbf{K}+\lambda T\mathbf{I})^{-1})$ *is the number of effective degrees of freedom that indicates the number of independent parameters in a learning problem [83].*

The above theorem expresses the trade-off between the computational efficiency and the statistical efficiency through the regularization parameter λ, effective dimension $d_{\mathbf{K}}^{\lambda}$, and the number of random features adopted for both regression and classification tasks using plain RF mapping. We derive the generation bound for the regression task while the bound for the binary classification task is adapted from [91, Corollary 4]. The bounds of random features can be further tightened by leveraging the data-dependent sampling strategies [91]. We can see that to bound the excess risk with a higher probability, we need more random features, which results in a higher computational complexity. The regularization parameter is usually determined by the number of training data. In particular, [84] shows that setting $\lambda = \mathcal{O}(1/\sqrt{T})$ achieves the minimax risk convergence rate for kernel ridge regression problem. In general, kernel learning is amenable to theoretical analysis, and its unification with NNs suggests a viable path to theoretical understanding of NNs.

Theorem 12. *(Universal approximation of DKL.) Let* $\mathcal{C} \neq \{0\}$ *be any fixed nonempty subset of* $\mathbb{R}^d$*, then, for any* $f \in L^2(\mathcal{C}), \delta \in (0,1)$*, and* $\epsilon > 0$*, there exists a positive integer* L_{M-1} *such that with probability greater than* $1-\delta$*, we can find coefficients* $\boldsymbol{\vartheta} \in \mathbb{R}^{L_{M-1}}$ *that satisfy*

$$\|\hat{f}_M - f\|_{L^2(\mathcal{C})} \leq \epsilon. \tag{5.13}$$

Proof. The proof relies on mathematic induction. Consider two positive definite kernels: $\kappa_1 : \mathcal{X}^2 \to \mathbb{R}$ and its RKHS $\mathcal{H}_1$ with mapping $\varphi_1 : \mathcal{X} \to \mathcal{H}_1$; and $\kappa_2 : \mathcal{H}_1^2 \to \mathbb{R}$ and its

RKHS $\mathcal{H}_2$ with mapping $\varphi_2 : \mathcal{H}_1 \to \mathcal{H}_2$. Then, [150] proves $\kappa_3 : \mathcal{X}^2 \to \mathbb{R}$ below is also positive definite

$$\kappa_3(\mathbf{x}, \mathbf{x}_t) = \kappa_2(\varphi_1(\mathbf{x}), \varphi_1(\mathbf{x}_t)), \tag{5.14}$$

and its RKHS mapping is $\varphi_3 = \varphi_2 \circ \varphi_1$. Therefore, by mathematic induction the composed kernel κ_{M-1} of $M-1$ layers is also positive definite if the deep kernel model is equipped with positive definite kernels across all layers. Specifically, for Gaussian kernels adopted in this paper, the composed kernel κ_{M-1} is also a Gaussian kernel and its corresponding RKHS $\mathcal{H}_{M-1}$ is dense, which indicates that $\mathcal{H}_{M-1}$ is universal [151]. Then, for $f \in \mathcal{H}_{M-1}$ and by the mapping equivalence between the Gaussian kernel and its random features, we conclude that f can be approximated by $\hat{f}_M$ with high probability by random choices of $\boldsymbol{\omega}_l^m$ from the distribution $p_{\kappa_m}(\boldsymbol{\omega})$, $\forall l, m$. ■

The following theorem shows the universality of (5.10).

Theorem 13. *(Universal approximation of DKL-MLP.) Let $\mathcal{C} \neq \{0\}$ be any fixed nonempty subset of $\mathbb{R}^d$, then, for any $f \in L^2(\mathcal{C}), \delta \in (0,1)$ and $\epsilon > 0$, there exists a positive integer $L_\Sigma = L_1 + \cdots + L_{M-1}$ such that with probability greater than $1-\delta$, we can find coefficients $\boldsymbol{\Theta} \in \mathbb{R}^{L_\Sigma}$ that satisfy*

$$\|\tilde{f}_M - f\|_{L^2(\mathcal{C})} \leq \epsilon, \tag{5.15}$$

where $\boldsymbol{\Theta} = [\boldsymbol{\vartheta}^{M-1}; \cdots; \boldsymbol{\vartheta}^2; \boldsymbol{\vartheta}^1] \in \mathbb{R}^{L_\Sigma}$.

Proof. To prove that (5.10) admits universality, we can view it as a multi-kernel learning problem. Theorem 5.7 in [150] states that the space induced by the summation of two reproducing kernels is also a RKHS. Hence, by mathematical induction, the summation of $M-1$ reproducing kernels also induces a RKHS. For our model where all kernels are Gaussian, the composed kernel is also Gaussian, and its corresponding RKHS is dense. Thus, by the mapping equivalence between the Gaussian kernel and its random Fourier features, we conclude that f can be approximated by $\tilde{f}_M$ with high probability by random choices of $\boldsymbol{\omega}_l^m$ that follows the distribution $p_{\kappa_m}(\boldsymbol{\omega})$, $\forall l, m$. ■

Remark 6 (No bad local minimum). With $\boldsymbol{\Theta}$ defined in Theorem 13, we can rewrite (5.10) as

$$\tilde{f}_{\boldsymbol{\Theta}}(\mathbf{x}) = \boldsymbol{\Theta}^\top \boldsymbol{\Phi}(\mathbf{x}), \tag{5.16}$$

where $\boldsymbol{\Phi}(\mathbf{x}) = [\boldsymbol{\phi}^{M-1}(\cdots(\boldsymbol{\phi}^1(\mathbf{x})))); \cdots; \boldsymbol{\phi}^2(\boldsymbol{\phi}^1(\mathbf{x})); \boldsymbol{\phi}^1(\mathbf{x})]$. Then, (5.1) becomes

$$\min_{\boldsymbol{\Theta}} \quad \hat{R}(\boldsymbol{\Theta}) := \frac{1}{T}\sum_{t=1}^{T} \ell(\tilde{f}_{\boldsymbol{\Theta}}(\mathbf{x}_t), y_t) + \lambda \|\boldsymbol{\Theta}\|_2^2. \tag{5.17}$$

It is clear that the globally optimal parameters can be found via convex optimization. Therefore, our model has no bad local minimum by Theorem 13.

5.6 Extension to Distributed Multi-agent Settings

Although this chapter focuses on single-agent learning, it can be easily extended to distributed learning over multi-agent systems. In fact, implementing RF-KL (5.8) at each agent with $\boldsymbol{\theta}$ being the model parameter that to be collaboratively learned is exactly the distributed RF-based kernel learning problem (2.16) discussed in Chapter 2. Similarly, for the DKL-MLP method, each agent i will be equipped with a local cost function 5.17 parameterized by parameter $\boldsymbol{\Theta}_i$, which is the local copy of $\boldsymbol{\Theta}$. Then these agents collaboratively solve an objective function which is the summation of all local cost functions with the consensus constraint enforced on neighboring agents, as given by (2.17). ADMM is then utilized for iterative and distributed implementation, communication censoring and quantization can then be utilized to save communication resources. A similar extension can be achieved in online and dynamic settings where the data is collected in a streaming manner with possibly unknown dynamics. Note that for the single-agent case, gradient descent is utilized to learn the model parameter, which can also be utilized in the distributed case. For distributed learning with a coordinate center, the central node plays a role to combine and average the model updates from all other nodes, and we can utilize the censoring rule

proposed in [10] to save communication. For a fully decentralized network, we can apply the combine-then-adapt scheme for information aggregation over the network, as proposed by [58] for decentralized online kernel learning. The communication censoring and quantization strategies can be embedded to increase communication efficiency, and theoretical analysis needs to be further investigated to ensure convergence. to ensure convergence.

5.7 Experiments

In this section, we conduct experiments to compare DKL-MLP with several benchmarks using real datasets from UCI repository [86] for regression and classification tasks, detailed descriptions of all datasets are listed in Table 5.2.

Table 5.2: Datasets details.

Dataset	training data	test data	data dimension	task
Twitter	10500	3500	77	regression
Tom's hardware	8250	2750	96	regression
Adult	31655	13567	103	classification
Sensor	40956	17553	48	classification
Human activity	5514	1838	30	classification
Letter	14000	6000	16	classification

RF-KL is the RF-based kernel method in the structure of a two-layer neural network, as described by the model (5.7).

DKL is the vanilla RF-based deep kernel learning network described by (5.9).

Deep-semi is the work proposed in [146]. For fair comparison, we utilize Gaussian kernels and their corresponding random Fourier features instead of the proposed Heaviside step function.

DNN is the vanilla deep neural network that all parameters need to be trained.

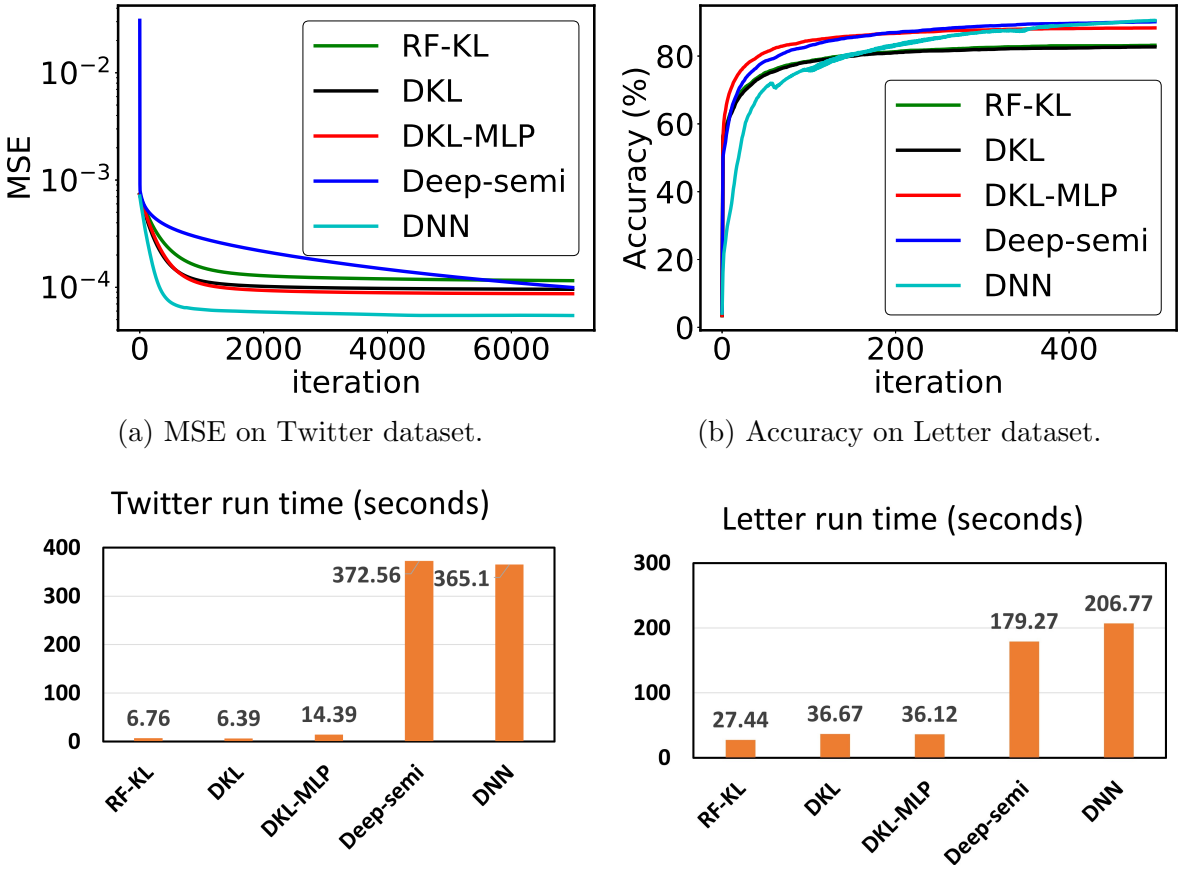
All kernel-based methods use Gaussian kernels with their best kernel bandwidth optimized through grid search for each dataset and for each algorithm, whereas DNN employs Relu functions for all hidden layers. Depending on the tasks, the output layer equips

with a sigmoid function, a softmax function, or directly outputs the results. All methods are trained using gradient descent with their learning rate optimized through grid search for each dataset for each method. For all tests, the regularization parameter is set to be $\lambda = 10^{-4}$ and we use 75% of the data for training and the remaining data for testing for all datasets.

In our simulations, since the regression and classification problems are not too complicated with mild-size datasets, we apply 2 hidden layers which are sufficient for good learning performance. We have conducted extensive simulations on these datasets and selected to present the results for one regression (Twitter) and one classification (Letter) problems using figures and the simulations results on the other datasets using tables considering the simulation results on the other datasets present the similar trend as shown in Figure 5.3.

Specifically, the Twitter dataset consists of $T = 14000$ samples with $\mathbf{x}_t \in \mathbb{R}^{77}$ being a feature vector reflecting the number of new interactive authors and the length of discussions on a given topic, etc., and $y_t \in \mathbb{R}$ representing the average number of active discussions on a certain topic. The learning task is to predict the popularity of these topics. Letter dataset consists of $T = 20000$ samples with $\mathbf{x}_t \in \mathbb{R}^{16}$ and y_t represents letters from A to Z. The learning task is to identify the capital letters. RF-KL has 50 neurons for the Twitter dataset and 200 for the Letter dataset. All other deep networks have 50 neurons in each hidden layer for the Twitter dataset and 200 neurons in each hidden layer for the Letter dataset. Figure 5.3 (a) and Figure 5.3 (b) show that eventually DNN and Deep-semi methods perform better than the other methods, which makes sense. Take the Twitter dataset as an example, Deep-semi has 6400 parameters and DNN has 6501 trainable parameters, while DKL-MLP only has 100 trainable parameters. Also, DKL-MLP performs better than DKL and RF-KL since the latter two only have 50 trainable parameters, respectively. The running time of all algorithms on the two datasets is presented in Figure 5.3 (c) and Figure 5.3 (d), and suggests that RF-based kernel methods enjoy low complexity. Simulation results on the other datasets are listed in Tables 5.3 5.6. These experimental results corroborate that DKL-MLP achieves a good trade-off of learning performance and computational complexity

compared with the benchmark methods. Note that if the training resource is limited, DNN and Deep-semi methods may not be able to train a large model and their representation power would degrade. On the other hand, if the training time is limited, which means DNN and Deep-semi methods may not have enough time to converge, their learning performance will also be affected.



(a) MSE on Twitter dataset.

(b) Accuracy on Letter dataset.

(c) Run time on Twitter dataset.

(d) Run time on Letter dataset.

Figure 5.3: Performance comparison.

Table 5.3: Performance comparison on Tom's hardware.

Algorithms	network structure	# parameters	MSE (10^{-4})	running time (s)
RF-KL	$L = 100$	100	1.94	14.92
	$L = 200$	200	1.61	33.01
DKL	$L_1 = L_2 = 100$	100	1.86	16.94
	$L_1 = L_2 = 200$	200	1.80	34.09
DKL-MLP	$L_1 = L_2 = 100$	200	1.68	35.07
	$L_1 = L_2 = 200$	400	1.54	67.9
Deep-semi	$L_1 = L_2 = 100$	19700	1.17	729.59
DNN	$L_1 = L_2 = 25$	3101	0.8	208

Table 5.4: Performance comparison on Sensor dataset.

Algorithms	network structure	# parameters	accuracy (%)	running time (s)
RF-KL	$L = 200$	200	90.83	42.4
	$L = 400$	400	93.27	61.7
DKL	$L_1 = L_2 = 200$	200	90.3	43.45
	$L_1 = L_2 = 400$	400	93.52	58.59
DKL-MLP	$L_1 = L_2 = 200$	400	94.41	60.86
	$L_1 = L_2 = 400$	800	96.09	97.12
Deep-semi	$L_1 = L_2 = 20$	1380	75.81	81.59
	$L_1 = L_2 = 200$	15400	97.07	599

Table 5.5: Performance comparison on Adult dataset.

Algorithms	network structure	# parameters	accuracy (%)	running time (s)
RF-KL	$L = 100$	100	73.66	8.86
	$L = 500$	500	75.1	32.92
DKL	$L_1 = L_2 = 100$	100	73.14	9.42
	$L_1 = L_2 = 500$	500	74.45	33.58
DKL-MLP	$L_1 = L_2 = 100$	200	75.1	16.27
	$L_1 = L_2 = 500$	1000	76.31	63.65
Deep-semi	$L_1 = L_2 = 100$	20400	79.51	439.09
DNN	$L_1 = L_2 = 100$	20601	79.98	391.89

5.8 Conclusion

In this chapter, we first build a unified framework of kernel learning and neural network by leveraging the random feature mapping method, where kernel methods can be implemented

Table 5.6: Performance comparison on Human activity dataset.

Algorithms	network structure	# parameters	accuracy (%)	running time (s)
RF-KL	$L = 10$	10	95.13	0.15
	$L = 20$	20	97.48	0.19
DKL	$L_1 = L_2 = 10$	10	94.37	0.15
	$L_1 = L_2 = 20$	20	97.99	0.19
DKL-MLP	$L_1 = L_2 = 10$	20	97.3	0.21
	$L_1 = L_2 = 20$	40	98.46	0.25
Deep-semi	$L_1 = L_2 = 10$	410	98.89	0.73
DNN	$L_1 = L_2 = 10$	431	98.92	1.07

using the same neural network structure. We then develop a deep kernel learning network with multiple learning paths. Leveraging the theory from traditional kernel methods, we prove that our proposed deep kernel learning networks admit universality. The learnability of our method can be improved by the trainable paths and the computational complexity is greatly reduced via random feature mapping. It should be noted that the multiple-learning-path scheme can also be applied to other deep networks such as deep extreme learning machines and deep neural networks.

Chapter 6: Summary and Future Research Directions

This chapter summarizes the research problems investigated in the present dissertation and proposes some potential research directions highly related to this dissertation.

6.1 Summary

This dissertation mainly tackles the communication efficiency problem in distributed learning over multi-agent systems, as well as makes the first step to bridge the gap between non-parametric kernel methods and neural networks. To summarize, our efforts are devoted to solving the following problems.

P1: How to develop distributed kernel learning algorithms under the consensus optimization framework that do not involve raw data transmission?

P2: How to ensure convergence/generalization performance as well as communication efficiency of the developed distributed algorithms?

P3: How should the distributed kernel algorithms developed for P1 be adapted to cope with online streaming data as well as streaming data with unknown dynamics?

P4: Can communication efficiency still be guaranteed at almost no cost of the learning performance for the online setting?

P5: What are the differences and connections between kernel methods and neural networks? Can they be combined to obtain a more powerful version that inherits the advantages of both methods while bypassing the disadvantages of them?

To solve these problems, we mainly utilize the following tools.

Table 6.1: Dissertation summary

Chapter	Problems	Tools	Algorithms	Theoretical analysis
2	P1, P2	T1, T2, T3	DKLA, COKE	Convergence, generalization
3	P3, P4	T1, T3, T4, T5	ODKLA, QC-ODKLA	Static regret
4	P3, P4	T1, T2, T3, T4	DDKL, QC-DDKL	Dynamic regret
5	P5	T1, T6	DKL, DKL-MLP	Universal approximation

T1: Random feature mapping: approximates kernel functions in the original data space to a random feature space.

T2: Alternating direction method of multipliers: a distributed implementation for consensus optimization problems with fast convergence rate.

T3: Communication-censoring strategy: evaluates if a message is informative enough to be transmitted.

T4: Quantization strategy: restricts the total number of bits transmitted in the learning process.

T5: Linearized alternating direction method of multipliers: a computation-light distributed implementation for consensus optimization problems.

T6: Multiple-learning-paths scheme: adds flexibility and representation power for deep kernel methods.

Table 6.1 summarizes the corresponding tools utilized for the research problems, the developed algorithms, and the theoretical foundations backing these algorithms.

6.2 Future Research Directions

Moving forward, we would like to investigate the following directions that are closely related to our current research.

- **Designing state-dependent censoring rules.** The simulation results in Chapter 2 - 4 validate the communication reduction of communication censoring rules. However,

how to adaptively design the censoring parameters remains a question. Based on our previous research on event-trigger control [19,20] and the state-dependent rule devised in [10], we propose to utilize the Lyapunov function related with the loss functions and state information of the networked agents to develop stat-dependent censoring rules so that agents can adaptively adjust their censoring parameters according to the network structure and their local information. Direct designing these rules for conventional ADMM may be challenging, we will start from linearized ADMM where a closed-form solution for the primal update exists and shows some similarities with the gradient descent method in [10].

- **Adopting dynamic linearized ADMM.** Chapter 4 utilizes conventional ADMM to solve the dynamic kernel learning problem, and it requires more computation than linearized ADMM since conventional ADMM needs to solve optimization problems for the primal update at each iteration. To reduce the computational complexity for real-time implementation, a direct way would be to implement the dynamic linearized ADMM.

- **Interpreting DNNs using kernel methods.** Chapter 5 proposes a unified learning framework for kernel methods and neural networks and proposes a DKL-MLP method to combine the benefits of both methods by adding multiple learning paths to the DKL structure. The combination should not be unique, and we would like to investigate other combination possibilities, such as implementing random feature mapping in the first several layers and training the remaining hidden layers using backpropagation as in standard neural networks. The DKL-MLP method proposed in Chapter 5 can be viewed as a deep multiple kernel learning problem, and if trained by the gradient descent method, we can use the algorithm unrolling technique to reveal the connection between it and the iterative algorithms used in signal processing. As such, it may also shed light on the interpretability of DNNs.

- **Going from kernel methods to Gaussian process and beyond.**The RF-based

distributed kernel methods proposed in this dissertation are efficient for distributed learning. However, they are not able to quantify uncertainty in learning problems. Uncertainty is important in machine learning since we need to know how confident we can be about a decision with the change environment and observations. Many applications of machine learning depend on accurate estimation of the uncertainty, such as forecasting, decision making, learning from limit, noisy, and missing data, as well as automating scientific modeling and experiment design. We propose to utilize the RF method for scalable distributed Gaussian process learning and to quantify the uncertainty of the function estimates. Some preliminary research on RF-based Gaussian process and distributed Gaussian process are available in [152, 153]. Going beyond, we can also explore the connections between DNNs and the deep Gaussian process to interpret DNNs [154]. Another direction can be devising distributed deep Gaussian process for function estimation. Apart from that, uncertainty-aware multi-agent reinforcement learning is also worth investigating [155, 156].

Bibliography

Bibliography

[1] W. Ren, R. W. Beard, and E. M. Atkins, "Information consensus in multivehicle cooperative control," *IEEE Control Systems Magazine*, vol. 27, no. 2, pp. 71–82, 2007.

[2] J. Liang, Z. Wang, B. Shen, and X. Liu, "Distributed state estimation in sensor networks with randomly occurring nonlinearities subject to time delays," *ACM Transactions on Sensor Networks (TOSN)*, vol. 9, no. 1, pp. 1–18, 2012.

[3] M. U. Ilyas, M. Z. Shafiq, A. X. Liu, and H. Radha, "A distributed algorithm for identifying information hubs in social networks," *IEEE Journal on Selected Areas in Communications*, vol. 31, no. 9, pp. 629–640, 2013.

[4] R. Bhardwaj, A. R. Nambiar, and D. Dutta, "A study of machine learning in healthcare," in *2017 IEEE 41st Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2. IEEE, 2017, pp. 236–241.

[5] G. V. Demarie and D. Sabia, "A machine learning approach for the automatic long-term structural health monitoring," *Structural Health Monitoring*, vol. 18, no. 3, pp. 819–837, 2019.

[6] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, vol. 3, 2017.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2017, pp. 1273–1282.

[8] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Third Workshop on Bayesian Deep Learning (NeurIPS)*, 2018.

[9] A. G. Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "Braintorrent: A peer-to-peer environment for decentralized federated learning," *arXiv preprint arXiv:1905.06731*, 2019.

[10] T. Chen, G. Giannakis, T. Sun, and W. Yin, "LAG: Lazily aggregated gradient for communication-efficient distributed learning," in *Advances in Neural Information Processing Systems*, 2018, pp. 5050–5060.

[11] X. Lian, W. Zhang, C. Zhang, and J. Liu, "Asynchronous decentralized parallel stochastic gradient descent," in *International Conference on Machine Learning (ICML)*. PMLR, 2018, pp. 3043–3052.

[12] T. Wu, K. Yuan, Q. Ling, W. Yin, and A. H. Sayed, "Decentralized consensus optimization with asynchrony and delays," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 2, pp. 293–307, 2017.

[13] T. Hofmann, B. Schölkopf, and A. J. Smola, "Kernel methods in machine learning," *The Annals of Statistics*, pp. 1171–1220, 2008.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[15] J. Shawe-Taylor, N. Cristianini *et al.*, *Kernel methods for pattern analysis*. Cambridge University Press, 2004.

[16] F. Pérez-Cruz and O. Bousquet, "Kernel methods and their potential use in signal processing," *IEEE Signal Processing Magazine*, vol. 21, no. 3, pp. 57–65, 2004.

[17] P. Xu, Z. Tian, Z. Zhang, and Y. Wang, "COKE: Communication-censored kernel learning via random features," in *2019 IEEE Data Science Workshop (DSW)*. IEEE, 2019, pp. 32–36.

[18] P. Xu, Y. Wang, X. Chen, and Z. Tian, "COKE: Communication-censored decentralized kernel learning," *Journal of Machine Learning Research*, vol. 22, no. 196, pp. 1–35, 2021.

[19] P. Xu, C. Nowzari, and Z. Tian, "A class of event-triggered coordination algorithms for multi-agent systems on weight-balanced digraphs," in *2018 Annual American Control Conference (ACC)*. IEEE, 2018, pp. 5988–5993.

[20] ——, "A class of distributed event-triggered average consensus algorithms for multi-agent systems," *International Journal of Control*, vol. 95, no. 2, pp. 502–515, 2020.

[21] P. Xu, Z. Tian, and Y. Wang, "An energy-efficient distributed average consensus scheme via infrequent communication," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2018, pp. 648–652.

[22] K. Slavakis, G. B. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics: (statistical) learning tools for our era of data deluge," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 18–31, 2014.

[23] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: an application of large-scale online learning," in *International Conference on Machine Learning (ICML)*. PMLR, 2009, pp. 681–688.

[24] P. Xu, Y. Wang, X. Chen, and Z. Tian, "QC-ODKLA: Quantized and communication-censored online decentralized kernel learning via linearized ADMM," *submitted to ICML 2022*.

[25] ——, "Communication-efficient decentralized dynamic kernel learning," *in preparation, aim to submit to NeurIPS 2023*.

[26] ——, "Deep kernel learning networks with multiple learning paths," in *2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*.

[27] W. Shi, Q. Ling, K. Yuan, G. Wu, and W. Yin, "On the linear convergence of the ADMM in decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 62, no. 7, pp. 1750–1761, 2014.

[28] A. Nedić, A. Olshevsky, and C. A. Uribe, "A tutorial on distributed (non-bayesian) learning: Problem, algorithms and results," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 6795–6801.

[29] J. B. Predd, S. R. Kulkarni, and H. V. Poor, "Distributed kernel regression: An algorithm for training collaboratively," in *2006 IEEE Information Theory Workshop (ITW)*. IEEE, 2006, pp. 332–336.

[30] R. Mitra and V. Bhatia, "The diffusion-KLMS algorithm," in *2014 International Conference on Information Technology*. IEEE, 2014, pp. 256–259.

[31] W. Gao, J. Chen, C. Richard, and J. Huang, "Diffusion adaptation over networks with kernel least-mean-square," in *2015 IEEE 6th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*. IEEE, 2015, pp. 217–220.

[32] S. Chouvardas and M. Draief, "A diffusion kernel LMS algorithm for nonlinear adaptive networks," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4164–4168.

[33] B.-S. Shin, H. Paul, and A. Dekorsy, "Distributed kernel least squares for nonlinear regression applied to sensor networks," in *2016 24th European Signal Processing Conference (EUSIPCO)*. IEEE, 2016, pp. 1588–1592.

[34] B.-S. Shin, M. Yukawa, R. L. G. Cavalcante, and A. Dekorsy, "Distributed adaptive learning with multiple kernels in diffusion networks," *IEEE Transactions on Signal Processing*, vol. 66, no. 21, pp. 5505–5519, 2018.

[35] A. Koppel, S. Paternain, C. Richard, and A. Ribeiro, "Decentralized online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 66, no. 12, pp. 3240–3255, 2018.

[36] S. Bucak, R. Jin, and A. K. Jain, "Multi-label multiple kernel learning by stochastic approximation: Application to visual object recognition," in *Advances in Neural Information Processing Systems*, vol. 23, 2010.

[37] B. Gu, M. Xin, Z. Huo, and H. Huang, "Asynchronous doubly stochastic sparse kernel learning," in *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[38] R. Gomes and A. Krause, "Budgeted nonparametric learning from data streams," in *International Conference on Machine Learning (ICML)*. PMLR, 2010, pp. 391–398.

[39] Z. Wang, K. Crammer, and S. Vucetic, "Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale SVM training," *Journal of Machine Learning Research*, vol. 13, pp. 3103–3131, 2012.

[40] L. Zhang, J. Yi, R. Jin, M. Lin, and X. He, "Online kernel learning with a near optimal sparsity bound," in *International Conference on Machine Learning (ICML)*. PMLR, 2013, pp. 621–629.

[41] T. Le, V. Nguyen, T. D. Nguyen, and D. Phung, "Nonparametric budgeted stochastic gradient descent," in *International Conference on Artificial Intelligence and Statistics*, 2016, pp. 564–572.

[42] A. Koppel, G. Warnell, E. Stump, and A. Ribeiro, "Parsimonious online learning with kernels via sparse projections in function space," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4671–4675.

[43] P. Honeine, "Analyzing sparse dictionaries for online learning with kernels," *IEEE Transactions on Signal Processing*, vol. 63, no. 23, pp. 6343–6353, 2015.

[44] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[45] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Transactions on Signal Processing*, vol. 57, no. 3, pp. 1058–1067, 2008.

[46] P. Drineas and M. W. Mahoney, "On the Nyström method for approximating a Gram matrix for improved kernel-based learning," *Journal of Machine Learning Research*, vol. 6, pp. 2153–2175, 2005.

[47] B. Dai, B. Xie, N. He, Y. Liang, A. Raj, M.-F. F. Balcan, and L. Song, "Scalable kernel methods via doubly stochastic gradients," in *Advances in Neural Information Processing Systems*, 2014, pp. 3041–3049.

[48] J. Lu, S. C. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu, "Large scale online kernel learning," *Journal of Machine Learning Research*, vol. 17, no. 1, pp. 1613–1655, 2016.

[49] F. Sheikholeslami, D. Berberidis, and G. B. Giannakis, "Large-scale kernel-based feature extraction via low-rank subspace tracking on a budget," *IEEE Transactions on Signal Processing*, vol. 66, no. 8, pp. 1967–1981, 2018.

[50] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Advances in Neural Information Processing Systems*, 2008, pp. 1177–1184.

[51] E. G. Băzăvan, F. Li, and C. Sminchisescu, "Fourier kernel learning," in *European Conference on Computer Vision*. Springer, 2012, pp. 459–473.

[52] T. D. Nguyen, T. Le, H. Bui, and D. Phung, "Large-scale online kernel learning with random feature reparameterization," in *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*. AAAI Press, 2017, pp. 2543–2549.

[53] F. X. X. Yu, A. T. Suresh, K. M. Choromanski, D. N. Holtmann-Rice, and S. Kumar, "Orthogonal random features," in *Advances in Neural Information Processing Systems*, 2016, pp. 1975–1983.

[54] Y. Shen, T. Chen, and G. B. Giannakis, "Online multi-kernel learning with orthogonal random features," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2018, pp. 6289–6293.

[55] F. Bach, "On the equivalence between kernel quadrature rules and random feature expansions," *Journal of Machine Learning Research*, vol. 18, no. 1, pp. 714–751, 2017.

[56] A. Rudi and L. Rosasco, "Generalization properties of learning with random features," in *Advances in Neural Information Processing Systems*, 2017, pp. 3215–3225.

[57] Z. Li, J.-F. Ton, D. Oglic, and D. Sejdinovic, "Towards a unified analysis of random Fourier features," *Journal of Machine Learning Research*, vol. 22, no. 108, 2021.

[58] P. Bouboulis, S. Chouvardas, and S. Theodoridis, "Online distributed learning over networks in RKH spaces using random Fourier features," *IEEE Transactions on Signal Processing*, vol. 66, no. 7, pp. 1920–1932, 2018.

[59] Y. Liu, W. Xu, G. Wu, Z. Tian, and Q. Ling, "Communication-censored ADMM for decentralized consensus optimization," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2565–2579, 2019.

[60] S. Zhu, M. Hong, and B. Chen, "Quantized consensus ADMM for multi-agent distributed optimization," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4134–4138.

[61] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "QSGD: Communication-efficient SGD via gradient quantization and encoding," in *Advances in Neural Information Processing Systems*, 2017, pp. 1709–1720.

[62] M. Zhang, L. Chen, A. Mokhtari, H. Hassani, and A. Karbasi, "Quantized frank-wolfe: Communication-efficient distributed optimization," *arXiv preprint arXiv:1902.06332*, 2019.

[63] S. U. Stich, J.-B. Cordonnier, and M. Jaggi, "Sparsified SGD with memory," in *Advances in Neural Information Processing Systems*, 2018, pp. 4447–4458.

[64] D. Alistarh, T. Hoefler, M. Johansson, N. Konstantinov, S. Khirirat, and C. Renggli, "The convergence of sparsified gradient methods," in *Advances in Neural Information Processing Systems*, 2018, pp. 5973–5983.

[65] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," in *Advances in Neural Information Processing Systems*, 2018, pp. 1299–1309.

[66] I. E. K. Harrane, R. Flamary, and C. Richard, "On reducing the communication cost of the diffusion LMS algorithm," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 5, no. 1, pp. 100–112, 2018.

[67] J. F. Mota, J. M. Xavier, P. M. Aguiar, and M. Püschel, "D-ADMM: A communication-efficient distributed algorithm for separable optimization," *IEEE Transactions on Signal Processing*, vol. 61, no. 10, pp. 2718–2723, 2013.

[68] M. Li, D. G. Andersen, A. J. Smola, and K. Yu, "Communication efficient distributed machine learning with the parameter server," in *Advances in Neural Information Processing Systems*, 2014, pp. 19–27.

[69] M. Jaggi, V. Smith, M. Takác, J. Terhorst, S. Krishnan, T. Hofmann, and M. I. Jordan, "Communication-efficient distributed dual coordinate ascent," in *Advances in Neural Information Processing Systems*, 2014, pp. 3068–3076.

[70] R. Arablouei, S. Werner, K. Doğançay, and Y.-F. Huang, "Analysis of a reduced-communication diffusion LMS algorithm," *Signal Processing*, vol. 117, pp. 355–361, 2015.

[71] W. Yin, X. Mao, K. Yuan, Y. Gu, and A. H. Sayed, "A communication-efficient random-walk algorithm for decentralized optimization," *arXiv preprint arXiv:1804.06568*, 2018.

[72] Y. Yu, J. Wu, and L. Huang, "Double quantization for communication-efficient distributed optimization," in *Advances in Neural Information Processing Systems*, 2019, pp. 4440–4451.

[73] W. Li, Y. Liu, Z. Tian, and Q. Ling, "COLA: Communication-censored linearized ADMM for decentralized consensus optimization," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5237–5241.

[74] O. Shamir, N. Srebro, and T. Zhang, "Communication-efficient distributed optimization using an approximate newton-type method," in *International Conference on Machine Learning (ICML)*. PMLR, 2014, pp. 1000–1008.

[75] S. J. Reddi, J. Konečnỳ, P. Richtárik, B. Póczós, and A. Smola, "Aide: Fast and communication efficient distributed optimization," *arXiv preprint arXiv:1608.06879*, 2016.

[76] B. Li, S. Cen, Y. Chen, and Y. Chi, "Communication-efficient distributed optimization in networks with gradient tracking and variance reduction," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 1662–1672.

[77] B. Schölkopf, R. Herbrich, and A. J. Smola, "A generalized representer theorem," in *International Conference on Computational Learning Theory*. Springer, 2001, pp. 416–426.

[78] H. Terelius, U. Topcu, and R. M. Murray, "Decentralized multi-agent optimization via dual decomposition," *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 11 245–11 251, 2011.

[79] X. Ji, C. Hou, Y. Hou, F. Gao, and S. Wang, "A distributed learning method for ℓ-1 regularized kernel machine over wireless sensor networks," *Sensors*, vol. 16, no. 7, p. 1021, 2016.

[80] S. Bochner, *Harmonic analysis and the theory of probability*. Courier Corporation, 2005.

[81] D. J. Sutherland and J. Schneider, "On the error of random Fourier features," *arXiv preprint arXiv:1506.02785*, 2015.

[82] B. Sriperumbudur and Z. Szabó, "Optimal rates for random Fourier features," in *Advances in Neural Information Processing Systems*, 2015, pp. 1144–1152.

[83] H. Avron, M. Kapralov, C. Musco, C. Musco, A. Velingker, and A. Zandieh, "Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees," in *International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 253–262.

[84] A. Caponnetto and E. De Vito, "Optimal rates for the regularized least-squares algorithm," *Foundations of Computational Mathematics*, vol. 7, no. 3, pp. 331–368, 2007.

[85] A. H. Sayed, "Adaptation, learning, and optimization over networks," *Foundations and Trends in Machine Learning*, vol. 7, no. ARTICLE, pp. 311–801, 2014.

[86] A. Asuncion and D. Newman, "UCI machine learning repository," 2007.

[87] F. Kawala, A. Douzal-Chouakria, E. Gaussier, and E. Dimert, "Prédictions d'activité dans les réseaux sociaux en ligne," 2013.

[88] L. M. Candanedo, V. Feldheim, and D. Deramaix, "Data driven prediction models of energy use of appliances in a low-energy house," *Energy and Buildings*, vol. 140, pp. 81–97, 2017.

[89] S. De Vito, E. Massera, M. Piga, L. Martinotto, and G. Di Francia, "On field calibration of an electronic nose for benzene estimation in an urban pollution monitoring scenario," *Sensors and Actuators B: Chemical*, vol. 129, no. 2, pp. 750–757, 2008.

[90] P. L. Bartlett and S. Mendelson, "Rademacher and Gaussian complexities: Risk bounds and structural results," *Journal of Machine Learning Research*, vol. 3, pp. 463–482, 2002.

[91] Z. Li, J.-F. Ton, D. Oglic, and D. Sejdinovic, "Towards a unified analysis of random Fourier features," *arXiv preprint arXiv:1806.09178*, 2018.

[92] M. Zinkevich, "Online convex programming and generalized infinitesimal gradient ascent," in *International Conference on Machine Learning (ICML)*. PMLR, 2003, pp. 928–936.

[93] S. Shalev-Shwartz *et al.*, "Online learning and online convex optimization," *Foundations and Trends® in Machine Learning*, vol. 4, no. 2, pp. 107–194, 2012.

[94] E. Hazan, A. Agarwal, and S. Kale, "Logarithmic regret algorithms for online convex optimization," *Machine Learning*, vol. 69, no. 2, pp. 169–192, 2007.

[95] N. N. Schraudolph, J. Yu, and S. Günter, "A stochastic quasi-newton method for online convex optimization," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2007, pp. 436–443.

[96] N. Chen, A. Agarwal, A. Wierman, S. Barman, and L. L. Andrew, "Online convex optimization using predictions," in *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, 2015, pp. 191–204.

[97] D. Mateos-Núnez and J. Cortés, "Distributed online convex optimization over jointly connected digraphs," *IEEE Transactions on Network Science and Engineering*, vol. 1, no. 1, pp. 23–37, 2014.

[98] H.-F. Xu, Q. Ling, and A. Ribeiro, "Online learning over a decentralized network through ADMM," *Journal of the Operations Research Society of China*, vol. 3, no. 4, pp. 537–562, 2015.

[99] A. Koppel, F. Y. Jakubiec, and A. Ribeiro, "A saddle point algorithm for networked online convex optimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 19, pp. 5149–5164, 2015.

[100] Y. Zhao, C. Yu, P. Zhao, H. Tang, S. Qiu, and J. Liu, "Decentralized online learning: Take benefits from others' data without sharing your own to track global trend," *arXiv preprint arXiv:1901.10593*, 2019.

[101] P. Sharma, P. Khanduri, L. Shen, D. J. Bucci Jr, and P. K. Varshney, "On distributed online convex optimization with sublinear dynamic regret and fit," *arXiv preprint arXiv:2001.03166*, 2020.

[102] M. Kamp, M. Boley, D. Keren, A. Schuster, and I. Sharfman, "Communication-efficient distributed online prediction by dynamic model synchronization," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2014, pp. 623–639.

[103] S. Shahrampour and A. Jadbabaie, "Distributed online optimization in dynamic environments using mirror descent," *IEEE Transactions on Automatic Control*, vol. 63, no. 3, pp. 714–725, 2017.

[104] S. M. Asghari, Y. Ouyang, and A. Nayyar, "Regret bounds for decentralized learning in cooperative multi-agent dynamical systems," in *Conference on Uncertainty in Artificial Intelligence*. PMLR, 2020, pp. 121–130.

[105] R. Dixit, A. S. Bedi, and K. Rajawat, "Online learning over dynamic graphs via distributed proximal gradient algorithm," *IEEE Transactions on Automatic Control*, vol. 66, no. 11, pp. 5065–5079, 2020.

[106] Y. Zhao, S. Qiu, K. Li, L. Luo, J. Yin, and J. Liu, "Proximal online gradient is optimum for dynamic regret: A general lower bound," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[107] K. Crammer, J. Kandola, and Y. Singer, "Online classification on a budget," *Advances in Neural Information Processing Systems*, vol. 16, 2003.

[108] Y. Shen, S. Karimi-Bidhendi, and H. Jafarkhani, "Distributed and quantized online multi-kernel learning," *IEEE Transactions on Signal Processing*, vol. 69, pp. 5496–5511, 2021.

[109] S. Hong and J. Chae, "Distributed online learning with multiple kernels," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[110] Q. Ling, W. Shi, G. Wu, and A. Ribeiro, "DLM: Decentralized linearized alternating direction method of multipliers," *IEEE Transactions on Signal Processing*, vol. 63, no. 15, pp. 4051–4064, 2015.

[111] W. Shi, Q. Ling, G. Wu, and W. Yin, "Extra: An exact first-order algorithm for decentralized consensus optimization," *SIAM Journal on Optimization*, vol. 25, no. 2, pp. 944–966, 2015.

[112] G. Qu and N. Li, "Harnessing smoothness to accelerate distributed optimization," *IEEE Transactions on Control of Network Systems*, vol. 5, no. 3, pp. 1245–1260, 2017.

[113] ——, "Accelerated distributed nesterov gradient descent," *IEEE Transactions on Automatic Control*, vol. 65, no. 6, pp. 2566–2581, 2019.

[114] Y. Liu, G. Wu, Z. Tian, and Q. Ling, "DQC-ADMM: Decentralized dynamic ADMM with quantized and censored communications," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[115] F. R. Chung and F. C. Graham, *Spectral graph theory*. American Mathematical Soc., 1997, no. 92.

[116] K. Hamidieh, "A data-driven statistical model for predicting the critical temperature of a superconductor," *Computational Materials Science*, vol. 154, pp. 346–354, 2018.

[117] M. Kachuee, M. M. Kiani, H. Mohammadzade, and M. Shabany, "Cuff-less high-accuracy calibration-free blood pressure estimation using pulse transit time," in *2015 IEEE international symposium on circuits and systems (ISCAS)*. IEEE, 2015, pp. 1006–1009.

[118] J. Sun, J. L. Moore, A. Bobick, and J. M. Rehg, "Learning visual object categories for robot affordance prediction," *The International Journal of Robotics Research*, vol. 29, no. 2-3, pp. 174–197, 2010.

[119] E. C. Hall and R. M. Willett, "Online convex optimization in dynamic environments," *IEEE Journal of Selected Topics in Signal Processing*, vol. 9, no. 4, pp. 647–662, 2015.

[120] O. Besbes, Y. Gur, and A. Zeevi, "Non-stationary stochastic optimization," *Operations research*, vol. 63, no. 5, pp. 1227–1244, 2015.

[121] A. Mokhtari, S. Shahrampour, A. Jadbabaie, and A. Ribeiro, "Online optimization in dynamic environments: Improved regret rates for strongly convex problems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*. IEEE, 2016, pp. 7195–7201.

[122] X. Cao and K. R. Liu, "Online convex optimization with time-varying constraints and bandit feedback," *IEEE Transactions on automatic control*, vol. 64, no. 7, pp. 2665–2680, 2018.

[123] T. Chen and G. B. Giannakis, "Bandit convex optimization for scalable and dynamic IoT management," *IEEE Internet of Things Journal*, vol. 6, no. 1, pp. 1276–1286, 2018.

[124] L. Zhang, T. Yang, Z.-H. Zhou *et al.*, "Dynamic regret of strongly adaptive methods," in *International conference on machine learning*. PMLR, 2018, pp. 5882–5891.

[125] L. Zhang, S. Lu, and T. Yang, "Minimizing dynamic regret and adaptive regret simultaneously," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 309–319.

[126] P. Zhao, Y.-J. Zhang, L. Zhang, and Z.-H. Zhou, "Dynamic regret of convex and smooth functions," *Advances in Neural Information Processing Systems*, vol. 33, pp. 12 510–12 520, 2020.

[127] Y. Zhang, R. J. Ravier, M. M. Zavlanos, and V. Tarokh, "A distributed online convex optimization algorithm with improved dynamic regret," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 2449–2454.

[128] X. Cao and T. Başar, "Decentralized online convex optimization with event-triggered communications," *IEEE Transactions on Signal Processing*, vol. 69, pp. 284–299, 2020.

[129] N. Eshraghi and B. Liang, "Improving dynamic regret in distributed online mirror descent using primal and dual information," *arXiv preprint arXiv:2112.03504*, 2021.

[130] Y. Shen, T. Chen, and G. B. Giannakis, "Random feature-based online multi-kernel learning in environments with unknown dynamics," *The Journal of Machine Learning Research*, vol. 20, no. 1, pp. 773–808, 2019.

[131] A. S. Bedi, A. Koppel, K. Rajawat, and B. M. Sadler, "Trading dynamic regret for model complexity in nonstationary nonparametric optimization," in *2020 American Control Conference (ACC)*. IEEE, 2020, pp. 321–326.

[132] G. Taylor, R. Burmeister, Z. Xu, B. Singh, A. Patel, and T. Goldstein, "Training neural networks without gradients: A scalable admm approach," in *International Conference on Machine Learning (ICML)*. PMLR, 2016, pp. 2722–2731.

[133] G. Yehudai and O. Shamir, "On the power and limitations of random features for understanding neural networks," in *Advances in Neural Information Processing Systems*, 2019, pp. 6598–6608.

[134] C. K. Williams, "Computing with infinite networks," in *Advances in neural information processing systems*, 1997, pp. 295–301.

[135] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Advances in neural information processing systems*, 2009, pp. 342–350.

[136] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari, "Linearized two-layers neural networks in high dimension," *arXiv preprint arXiv:1904.12191*, 2019.

[137] ——, "When do neural networks outperform kernel methods?" *arXiv preprint arXiv:2006.13409*, 2020.

[138] Z. Li, J.-F. Ton, D. Oglic, and D. Sejdinovic, "Towards a unified analysis of random Fourier features," in *International Conference on Machine Learning (ICML)*. PMLR, 2019, pp. 3905–3914.

[139] M. Jiu and H. Sahbi, "Nonlinear deep kernel learning for image annotation," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1820–1832, 2017.

[140] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Stochastic variational deep kernel learning," *arXiv preprint arXiv:1611.00336*, 2016.

[141] L. Le, J. Hao, Y. Xie, and J. Priestley, "Deep kernel: learning kernel function from data using deep neural network," in *Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, 2016, pp. 1–7.

[142] A. G. Wilson, Z. Hu, R. Salakhutdinov, and E. P. Xing, "Deep kernel learning," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2016, pp. 370–378.

[143] P.-S. Huang, L. Deng, M. Hasegawa-Johnson, and X. He, "Random features for kernel deep convex network," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 3143–3147.

[144] S. Mehrkanoon and J. A. Suykens, "Deep hybrid neural-kernel networks using random Fourier features," *Neurocomputing*, vol. 298, pp. 46–54, 2018.

[145] J. Xie, F. Liu, K. Wang, and X. Huang, "Deep kernel learning via random Fourier features," *arXiv preprint arXiv:1910.02660*, 2019.

[146] K. Kawaguchi, B. Xie, and L. Song, "Deep semi-random features for nonlinear function approximation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[147] K. Fang, X. Huang, F. Liu, and J. Yang, "End-to-end kernel learning via generative random Fourier features," *arXiv preprint arXiv:2009.04614*, 2020.

[148] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1-3, pp. 489–501, 2006.

[149] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[150] V. I. Paulsen and M. Raghupathi, *An introduction to the theory of reproducing kernel Hilbert spaces*. Cambridge university press, 2016, vol. 152.

[151] C. A. Micchelli, Y. Xu, and H. Zhang, "Universal kernels." *Journal of Machine Learning Research*, vol. 7, no. 12, 2006.

[152] Q. Lu, G. Karanikolas, Y. Shen, and G. B. Giannakis, "Ensemble Gaussian processes with spectral features for online interactive learning with scalability," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2020, pp. 1910–1920.

[153] M. Deisenroth and J. W. Ng, "Distributed Gaussian processes," in *International Conference on Machine Learning (ICML)*. PMLR, 2015, pp. 1481–1490.

[154] K. Cutajar, E. V. Bonilla, P. Michiardi, and M. Filippone, "Random feature expansions for deep Gaussian processes," in *International Conference on Machine Learning (ICML)*. PMLR, 2017, pp. 884–893.

[155] W. R. Clements, B. Van Delft, B.-M. Robaglia, R. B. Slaoui, and S. Toth, "Estimating risk and uncertainty in deep reinforcement learning," *arXiv preprint arXiv:1905.09638*, 2019.

[156] K. D. Polyzos, Q. Lu, A. Sadeghi, and G. B. Giannakis, "On-policy reinforcement learning via ensemble Gaussian processes with application to resource allocation," in *2021 55th Asilomar Conference on Signals, Systems, and Computers*. IEEE, pp. 1018–1022.

Curriculum Vitae

Ping Xu received her B.S. degree (with highest honors) in Electronic and Information Engineering from Northwestern Polytechnical University in 2015, and M.S. degree in Electrical Engineering from George Mason University in 2018. Her research interests span the areas of signal processing, wireless communications, cooperative control, network optimization, and machine learning, along with their applications in social, environmental, and IoT systems.