Scalable Learning Adaptive to Unknown Dynamics and Graphs

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE GRADUATE SCHOOL
OF THE UNIVERSITY OF MINNESOTA
BY

Yanning Shen

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

Prof. Georgios B. Giannakis, Advisor

June, 2019

Acknowledgements

First and foremost, my deepest gratitude goes to my Ph. D. advisor Prof. Georgios B. Giannakis. I would like to thank him for giving me the opportunity to embark on this journey as a graduate student, a real privilege for which I am really honored. His guidance and constant encouragement has made me become not only a better researcher, but also a better person. This thesis would not have been possible without his insightful suggestions. I would like to extend my appreciation to Professors Mostafa Kaveh, Mehmet Akçakaya, and George Karypis who agreed to take time out of their busy schedules and serve on my doctoral committee.

Throughout my graduate studies, I had the opportunity to collaborate with may talented individuals and I greatly benefited from their vision, ideas, and insights. I would like to extend my gratitude to Dr. Brian Baingana, Tianyi Chen, Dr. Morteza Mardani, Prof. Xiao Fu, Prof. Geert Leus, Prof. Nikolaos Sidiropoulos, Prof. Qing Ling, Prof. Sergio Barbarossa, Vassilis Ioannidis, Elena Ceci, Panagiotis Traganitis and Georgios Vasileios Karanikolas. The material in this thesis has also benefited from discussions with current and former members of the SPiNCOM group at UMN: Dimitris Berberidis, Seth Barrash, Emiliano Dall'Anese, Vassilis Ioannidis, Donghoon Lee, Bingcong Li, Meng Ma, Prof. Gonzalo Mateos, Prof. Daniel Romero, Alireza Sadeghi, Fatemeh Sheikholeslami, Prof. Konstantinos Slavakis, Dr. Yunlong Wang, Liang Zhang, and Prof. Yu Zhang. I am truly grateful to these people for their continuous help. I would also wish to acknowledge the grants that support financially our research.

I also want to express my thanks to my great friends, some of which I've already mentioned above, Cheng Jin, Agoritsa Polyzou, Ioanna Polyzou, Evangelia Christakopoulou, Konstantina Christakopoulou, Bo Yang, and Ahmed S. Zamzam for making my Ph.D. life a wonderful experience. My family has also been the greatest source of inspiration along this journey. Special thanks to my parents for their constant love, patience, care, and unwavering belief in me. And for my grandparents for their understanding and support for me to pursue my dream.

Yanning Shen, Minneapolis, December 11 2018.

Dedication

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

Abstract

With the scale of information growing every day, the key challenges in machine learning include the high-dimensionality and sheer volume of feature vectors that may consist of real and categorical data, as well as the speed and the typically streaming format of data acquisition that may also entail outliers and misses. The latter may be present, either unintentionally or intentionally, in order to cope with scalability, privacy, and adversarial behavior. These challenges provide ample opportunities for algorithmic and analytical innovations in online and nonlinear subspace learning approaches. Among the available nonlinear learning tools, those based on kernels have merits that are well documented. However, most rely on a preselected kernel, whose prudent choice presumes task-specific prior information that is generally not available. It is also known that kernel-based methods do not scale well with the size or dimensionality of the data at hand. Besides data science, the urgent need for scalable tools is a core issue also in network science that has recently emerged as a means of collectively understanding the behavior of complex interconnected entities. The rich spectrum of application domains comprises communication, social, financial, gene-regulatory, brain, and power networks, to name a few. Prominent tasks in all network science applications are those of topology identification and inference of nodal processes evolving over graphs. Most contemporary graph-driven inference approaches rely on linear and static models that are simple and tractable, but also presume that the nodal processes are directly observable.

To cope with these challenges, the present thesis first introduces a novel online categorical subspace learning approach to track the latent structure of categorical data 'on the fly.' Leveraging the random feature approximation, it then develops an adaptive online multi-kernel learning approach (termed AdaRaker), which accounts not only for data-driven learning of the kernel combination, but also for the unknown dynamics. Performance analysis is provided in terms of both static and dynamic regrets to quantify the novel learning function approximation. In addition, the thesis introduces a kernel-based topology identification approach that can even account for nonlinear dependencies among nodes and across time. To cope with nodal processes that may not be directly observable in certain applications, tensor-based algorithms that leverage piecewise stationary statistics of nodal processes are developed, and pertinent identifiability conditions are established. To facilitate real-time operation and inference of time-varying networks, an adaptive tensor decomposition based scheme is put forth to track the topologies of time-varying networks. Last but not least, the present thesis offers a unifying framework to deal with various learning tasks over possibly dynamic networks. These tasks include dimensionality reduction, classification, and clustering. Tests on both synthetic and real datasets from the aforementioned application domains are carried out to showcase the effectiveness of the novel algorithms throughout.

Contents

Acknowledgements **i**

Dedication **ii**

Abstract **iii**

List of Tables **viii**

List of Figures **x**

1 Introduction **1**

1.1 Motivation and context 1

1.1.1 Scalable online learning for big data 2

1.1.2 Network topology identification and tracking 2

1.1.3 Scalable graph-adaptive learning 3

1.2 Thesis outline and published results 4

1.2.1 Scalable subspace learning for big streaming categorical data 4

1.2.2 Online nonlinear learning in environments with unknown dynamics . . 4

1.2.3 Tensor-based network topology identification 5

1.2.4 Nonlinear network topology identification 5

1.2.5 Scalable nonlinear learning over graphs 6

1.3 Notational conventions 6

2 Scalable subspace learning for big Streaming Categorical data **7**

2.1 Introduction 7

2.2 Preliminaries and Problem Statement 9
2.2.1 Blind Probit model 10
2.2.2 Blind Tobit model 11
2.2.3 Blind Logit model 11
2.3 Rank-Regularized ML Estimation 12
2.4 Online Categorical Subspace Learning 15
2.4.1 First-order alternating minimization algorithms 16
2.4.2 Learning the quantizer 20
2.5 Performance Analysis 21
2.5.1 Asymptotic convergence analysis 21
2.5.2 Regret analysis 23
2.6 Numerical Tests 27
2.6.1 Synthetic data 28
2.6.2 Classification of chess games 30
2.6.3 Interpolation of “Movie-Lens” dataset 32
2.6.4 Threshold adaptation 33
2.7 Conclusions and Future Directions 34
2.8 Appendix 36
2.8.1 Proof of Lemma 1 36
2.8.2 Proof of Lemma 2 37
2.8.3 Proof of Lemma 3 39

3 Online nonlinear learning in environments with unknown dynamics 41
3.1 Introduction 41
3.1.1 Related works 42
3.1.2 Our contributions 43
3.2 Preliminaries and Problem Statement 44
3.3 Online MKL in static environments 45
3.3.1 RF-based single kernel learning 45
3.3.2 Raker for online MKL 48
3.3.3 Static regret analysis of Raker 52
3.4 Online MKL in Environments with Unknown Dynamics 54

3.4.1 AdaRaker with hierarchical ensembles 54
3.4.2 Dynamic regret analysis of AdaRaker 57
3.5 Numerical Tests 60
3.5.1 Synthetic data tests for regression 61
3.5.2 Real data tests for online regression 63
3.5.3 Real data tests for online classification 69
3.6 Summary 72
3.7 Appendix 73
3.7.1 Proof of Lemma 4 73
3.7.2 Proof of Theorem 1 76
3.7.3 Proof of Lemma 5 78
3.7.4 Proof of Theorem 2 79
3.7.5 Proof of Theorem 3 80

4 Tensor-based network topology identification 82

4.1 Introduction 82
4.2 Preliminaries and Problem Statement 85
4.3 A Tensor Factorization Approach 86
4.4 Identifiability issues 90
4.4.1 Identifiability with fully known $\mathbf{R}^x$ 91
4.4.2 Identifiability with partially known $\mathbf{R}^x$ 93
4.5 Tracking dynamic network topologies 95
4.5.1 Piecewise-invariant dynamic network topologies 95
4.5.2 Exponentially-weighted least-squares estimator 96
4.6 Numerical Tests 98
4.6.1 Tests on static simulated networks 99
4.6.2 Simulated piecewise-constant network 102
4.6.3 Tests on real networks 104
4.7 Summary 107
4.8 Appendix 108
4.8.1 Proof of Proposition 3 108
4.8.2 Proof of Lemma 8 108

4.8.3 Proof of Lemma 9 109
4.8.4 Proof of Lemma 10 109

5 Nonlinear network topology identification **111**
5.1 Introduction 111
5.2 Preliminaries on Linear SVARMs 113
5.3 From linear to nonlinear SVARMs 115
5.4 Kernel-based Sparse SVARMs 117
5.5 Data-driven kernel selection 120
5.6 Numerical tests 122
5.6.1 Synthetic data tests 122
5.6.2 Real data tests 125
5.7 Summary 131
5.8 Appendix 132
5.8.1 Topology Inference via ADMM 132

6 Online Graph-Adaptive Learning with Scalability and Privacy **135**
6.1 Introduction 135
6.2 Kernel-based learning over graphs 137
6.3 Online kernel-based learning over graphs 140
6.3.1 Batch RF-based learning over graphs 141
6.3.2 Online RF-based learning over graphs 142
6.4 Online Graph-adaptive MKL 144
6.4.1 Generalizations 146
6.5 Performance analysis 148
6.6 Numerical tests 150
6.6.1 Synthetic data test 151
6.6.2 Real data tests 153
6.7 Summary 155

References **156**

List of Tables

2.1 Runtime (seconds) and classification error comparison of the proposed online CSL scheme against the batch MM for synthetic data under variable fraction of misses $1 - p$. 30

2.2 Runtime and MAE comparison of the proposed scheme against the batch MM scheme under various p and different number of epochs for the Movie-Lens dataset with $d = 6$. 32

2.3 MAE and classification accuracy comparison of the novel CSL scheme with, and without threshold adaptation, under various p for binary synthetic data when $d = 5$, $D = 20$, and $T = 5,000$. 34

2.4 MAE and classification accuracy comparison of the CSL scheme with, and without threshold adaptation, under various p for the chess-game dataset when $d = 5$, $D = 35$, and $T = 3,196$. 35

3.1 Intervals and $\{\sigma^2\}$ for synthetic dataset. 61

3.2 CPU time (in seconds) on synthetic datasets. RBF, POLY represents all single-kernel methods using RBF and polynomial kernels, since they have the same CPU time. 64

3.3 A summary of real datasets used in the tests. 64

3.4 MSE (10^{-3}) performance of different algorithms with stepsize $1/\sqrt{T}$. 65

3.5 MSE (10^{-3}) performance of different algorithms with optimally chosen stepsizes. 66

3.6 MSE (10^{-3}) versus the choice of stepsizes with complexity $B = D = 50$. . . . 66

3.7 A summary of CPU time (second) on real datasets. 67

3.8 MSE (10^{-3}) versus complexity. For OMKL-B, the complexity measure is the data budget B; and for (Ada)Raker, the complexity measure is the number of RFs D. 68

3.9 CPU time (second) versus complexity of B for OMKL-B, and D for (Ada)Raker. 68
3.10 Classification error (%) and runtime (second) of different algorithms with the default stepsize $1/\sqrt{T}$ for RBF, OMKL and Raker, and with complexity $B = D = 50$. 69
3.11 Classification error (%) of different algorithms with the dataset-specific optimally chosen stepsizes for RBF, OMKL and Raker, and with complexity $B = D = 50$. 70
3.12 Classification error (%) versus different choices of stepsizes with $B = D = 50$. 70
3.13 Classification error (%) and CPU time (second) versus complexity. 71
4.1 Frequency of inference of networks depicted in Figure 4.12 out of 100 independent runs of Algorithm 5b on different segments of data with window lengths $L = 100$. 106
4.2 Frequency of inference of networks depicted in Figure 4.12 out of 100 independent runs of Algorithm 5b on the entire data set with different window lengths. 107
5.1 Comparison of global metrics associated with networks inferred from ECoG seizure data using the linear, K-SVARM, and K-SVARM with kernel selection scheme. Major differences between the computed metrics indicate that one may gain insights from network topologies inferred via models that capture nonlinear dependencies. 131

List of Figures

1.1 Examples of categorical datasets 2

1.2 Challenges of learning over networks 3

2.1 Illustration of the considered models, namely Probit, Tobit and Logit. 10

2.2 Empirical gradient-norm of (P3) versus time for synthetic data under variable $\%$ of misses $(1-p)$. 28

2.3 Time evolution of the two-dimensional (normalized) sketch obtained using the Probit model at (a) $t = 30$; (b) $t = 300$; and, (c) $t = 3,000$, when $d = 2$, and $D = 15$. The 'x' and 'o' markers are for two different classes. As time goes by, the classes become more separable. 29

2.4 (a) MAE and (b) SVM classification error of CSL scheme with Tobit versus the conventional PCA under variable compression rate p when $d = 20$, $D = 100$, and $T = 200$. 31

2.5 (a) Runtime and (b) LS classification error of the novel CSL scheme versus the MM scheme for the "King-Rook versus King-Pawn" dataset under variable dimension d when $p = 0.1$, $D = 35$, and $T = 3,196$. 31

2.6 MAE versus epochs under different settings of step size. 32

2.7 Quantization threshold convergence; (left) threshold evolution, and (right) threshold gradient absolute value evolution for chess data. 33

3.1 Hierarchical AdaRaker structure. Experienced experts in the middle layer present a Raker instance, where the size of expert cartoons is proportional to the interval length. 55

3.2 AdaRaker as an ensemble of Rakers with different learning rates: Each light/dark black interval initiates a Raker learner. At slot 7, colored experts are active, and gray ones are inactive. 57

3.3 MSE performance on synthetic Dataset 1: a) $D = B = 20$; b) $D = B = 50$. . . 62
3.4 MSE performance on synthetic Dataset 2: a) $D = B = 20$; b) $D = B = 50$. . . 63
4.1 An N-node directed network (blue links), with the t-th samples of endogenous measurements per node. SEMs explicitly account for exogenous inputs (red arrows), upon which endogenous variables may depend, in addition to the underlying topology. 85
4.2 The tensor constructed by stacking the correlation matrices admits a PARAFAC decomposition comprising rank-one tensor outer products. 89
4.3 Tensor grows per window m by a new frontal slice. 96
4.4 Actual and inferred adjacency matrices with the number of windows set to $M = 5, 10$, and 20. 100
4.5 EIER under varying window lengths M, and $\sigma_e = 0.1$: a) $\Omega = \{(i,j)|i = 1,\ldots,N, j = 1,\ldots,M\}$; b) 50% misses in $\mathbf{R}^x$; and c) $\Omega = \emptyset$. 100
4.6 EIER for several values of M and $\sigma_e = 1$: a) $\Omega = \{(i,j)|i = 1,\ldots,N, j = 1,\ldots,M\}$; b) 50% misses in $\mathbf{R}^x$; and c) $\Omega = \emptyset$. 101
4.7 Performance in blind scenario: a) EIER; b) Success rate. 102
4.8 EIER vs. m for: (a) Scenario p1; and (b) Scenario p2. 103
4.9 MSE vs m for: a) Scenario p1; b) Scenario p2. 103
4.10 Actual and inferred networks at $m = 200$. 104
4.11 Plot of the two groups of stock prices over the observation duration with zero-mean centering: a) technology companies; and b) online and "brick-and-mortar" retailers. The stock ticker symbol for each company is shown in the legend (in parentheses). 104
4.12 Visualization of network topologies inferred from the stock price time series, depicting: a) technology companies; and b) online and "brick-and-mortar" retailers. Notice the stronger dependencies between the two competing "brick-and-mortar" retailers, Macy's (MCY) and Nordstrom (NDM). 105

5.1 (left) A simple illustration of a 5-node brain network; and (right) a set of five neuronal time series (e.g., ECoG voltage) each associated with a node. Per interval t, SVARMs postulate that causal dependencies between the 5 nodal time series may be due to both the instantaneous effects (blue links), and/or time-lagged effects (red links). Estimating the values of the unknown coefficients amounts to learning the causal (link) structure of the network. 114
5.2 Plot of EIER vs. measurement ratio (T/N), with simulated data generated via a polynomial kernel of order $P = 2$. Note that K-SVARMs consistently outperform LSVARMs. 123
5.3 Plot of EIER vs. (T/N) with data generated using a Gaussian kernel with $\sigma^2 = 1$; K-SVARMs uniformly lead to lower errors than linear LSVARMs over varying SNR levels, based on empirical observations. 124
5.4 Plot of EIER vs. (T/N) with simulated data generated via a polynomial kernel of order $P = 2$; it can be empirically observed that K-SVARMs uniformly outperform LSVARMs across varying edge densities. 124
5.5 ROC curves generated under different modeling assumptions: a) K-SVARM based on a Gaussian kernel with $\sigma^2 = 1$; b) K-SVARM based on polynomial kernel of order $P = 2$; and c) Linear SVARM. 125
5.6 Visualizations of networks inferred from ECoG data: (a) linear SVARM with $L = 1$ on preictal time series; (b) linear SVARM on ictal time series; (c) K-SVARM on preictal time series, using Gaussian kernel with $\sigma = 1$; (d) the same K-SVARM on ictal time series; (e) K-SVARM with kernel selection on preictal time series; and finally (f) K-SVARM with kernel selection on ictal time series. 127
5.7 Node degrees of networks inferred from ECoG data encoded by circle radii: (a) linear SVARM on preictal data; (b) linear SVARM on ictal data; (c) K-SVARM on preictal time series; (d) K-SVARM on ictal data; (e) MKL-SVARM on preictal time series; (f) MKL-SVARM on ictal time series. 129
5.8 Same as in Figure 5.7 for comparison based on average shortest path length of inferred graphs. 130
6.1 Inference performance versus number of nodes for synthetic dataset generated from graph diffusion kernel 150

6.2 Inference performance versus number of nodes for synthetic dataset generated from Gaussian kernel 151
6.3 Inference performance versus number of sampled nodes in temperature dataset 152
6.4 Inference performance versus number of sampled nodes in email dataset 153
6.5 Inference performance versus number of sampled nodes in Cora dataset 154

Chapter 1

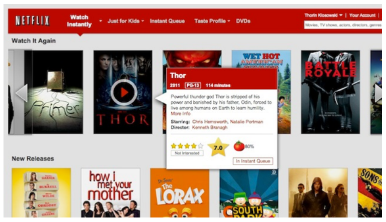
Introduction

1.1 Motivation and context

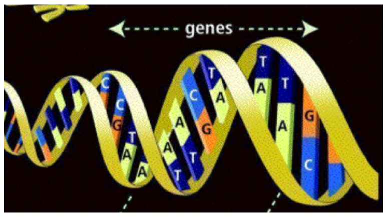
We live in an era of data deluge. Pervasive media collect massive amounts of data in a wide variety of formats. A major volume of this data originates from large-scale networks, which can represent a wide range of physical, biological, and social phenomena. For instance, we as users of the Facebook social network happily feed 10 billion messages per day, click the ‘like’ button 4.5 billion times, and upload 350 million new pictures each and every day. Learning from these large volumes of network data is expected to bring significant science and engineering advances along with consequent improvements in quality of life.

Networks are described by graphs, and identifying their topologies as well as processes evolving over graphs emerge in various applications involving brain, gene-regulatory, and social networks, to name a few. Knowing how graph nodes are connected is essential for understanding the dependencies among nodes in the underlying networks. Albeit simple and tractable, linear time-invariant models are limited since they are incapable of handling generally evolving topologies, as well as nonlinear and dynamic dependencies between nodal processes. To this end, one of the main high-level goals of this dissertation is to develop a unified framework capturing *nonlinearities* and *dynamics* present in real-world networks.

The key outcomes of this dissertation are algorithms, analysis, and application of machine learning and statistical signal processing tools to big data analytics, including scalable nonlinear learning from high-dimensional (network) data; which finds exciting applications in understanding the structure and dynamics of social, biological, and financial systems.



(a) Movie ratings



(b) Gene expressions

Figure 1.1: Examples of categorical datasets

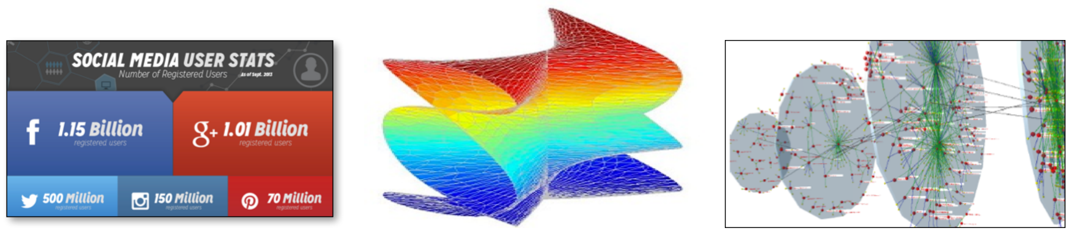
1.1.1 Scalable online learning for big data

Principal component analysis (PCA) is arguably the most popular tool for dimensionality reduction, with numerous applications in science and engineering [1]. It is however primarily designed to sketch high-dimensional data with analog-amplitude values, and does not apply to categorical data emerging for instance, with recommender systems; see also Fig. 1.1. Categorical PCA seeks a low-dimensional sketch of the high-dimensional categorical data to render affordable downstream machine learning tasks such as imputation, classification, and clustering; see e.g., [2–8]. However, the growing scale of nowadays 'Big Data' applications, such as recommender systems (e.g., NetFlix) with millions of users rating thousands of movies, pose extra challenges: (c1) the sheer volume of data approaches the computational and storage limits; (c2) new releases demand real-time processing for recommendations; and (c3) absent data entries, corresponding to missing user ratings. These challenges motivate well the first theme of this thesis on online nonlinear subspace learning.

Kernel-based methods exhibit well-documented performance in various nonlinear learning tasks. Major challenges of such methods include scalability and the choice of the kernel function, which presumes task-specific prior information. These considerations prompt algorithm development and analysis of scalable, adaptive, and data-driven multi-kernel approaches to track the unknown nonlinear learning function 'on the fly.'

1.1.2 Network topology identification and tracking

The study of networks and interconnections among complex agents has recently emerged as a major catalyst for collectively understanding the behavior of complex systems [9]. Such systems are ubiquitous, and commonly arise in both natural and man-made settings. For example, online interactions over the web are commonly facilitated through social networks such as Facebook



(a) Massive-scale (b) Unknown nonlinearity (c) Unknown dynamics

Figure 1.2: Challenges of learning over networks

and Twitter, while sophisticated brain functions are the result of vast interactions over complex neuronal networks. Other networks naturally emerge in settings as diverse as financial markets, genomics and proteomics, power grids, and transportation systems, to name just a few.

Topology identification of directed networks is a prominent task in the aforementioned application domains. For example, discovery of causal links between regions of interest in the brain is tantamount to identifying an implicit connectivity network. Studies pertaining to regulatory interactions among genes depend upon identification of unknown links within gene-regulatory networks. Since network structures are often unobservable, in order to facilitate network analytics, one generally resorts to approaches capitalizing on measurable nodal processes to infer the unknown topology. Most contemporary graph topology identification schemes rely on *linear* and *static* models due to their inherent simplicity. However, in complex systems such as gene or brain networks, these assumptions may not be feasible. This calls for nonlinear models and scalable online algorithms to cope with these challenges; see also Figure 1.2.

1.1.3 Scalable graph-adaptive learning

Estimating functions or signals specified over graph nodes is a task emerging in all the aforementioned network science applications. Functions of nodes can represent certain attributes or classes of these nodes. In Facebook for instance, each node represents a person, and the presence of an edge can indicate that two persons are friends, while nodal attributes can be age, gender or movie ratings of each person. In financial networks, where each node is a company, with links denoting trade between two companies, the function of the node can represent the category that each company belongs to, e.g., technology-, fashion-, or education-related.

In certain applications however, the size of the network may be very large, and new nodes may join the network over time. For example, hundreds of new users are joining Facebook or Netflix every day, and new companies are founded in financial networks regularly. Real-time and scalable estimation of the desired functions on these newly-joining nodes is of great importance. Besides scalability, nodes may have firm *privacy* requirements, and may therefore not be willing to reveal who their neighbors are. Most of the existing graph-aware learning methods however, generally require exact information of the network topology information, and therefore cannot meet privacy constraints. To this end, one of the major goals of this thesis is to develop a scalable graph-adaptive learning method with privacy.

1.2 Thesis outline and published results

This dissertation deals with scalable and adaptive learning for big data, topology identification and learning over graphs. Specifically, the outline and related publications are as follows.

1.2.1 Scalable subspace learning for big streaming categorical data

Common characteristics of large-scale datasets include the fact that they are often incomplete, prone to outlying measurements and may contain categorical attributes. Furthermore, as information sources unceasingly produce data in real time, analytics must often be performed on-the-fly, typically without a chance to reconsider previous data. These considerations justify why extracting latent low-dimensional structure from high-dimensional data is of paramount importance in timely inference tasks encountered with big data. The sheer volume of data and the fact that observations are acquired sequentially over time, motivate updating previously obtained analytics rather than recomputing new ones from scratch each time a new datum becomes available. In this context, Chapter 2 advocates a new framework to efficiently track the latent low-dimensional structures from incomplete and corrupt datasets typically encountered in practice. The proposed framework encompasses several fundamental learning tasks including imputation, clustering, and classification. Numerical tests for real MovieLens dataset and MINST dataset confirm the power of the novel methods compared with existing methods [10–12].

1.2.2 Online nonlinear learning in environments with unknown dynamics

Kernel-based methods exhibit well-documented performance in various nonlinear learning tasks. They are mainly challenged by the so-termed ‘curse of scalability,’ and the choice of the kernel function, which presumes task-specific prior information. Aiming to address these challenges, Chapter 3 introduces a scalable adaptive *data-driven* multi-kernel learning scheme to obtain the sought nonlinear learning function 'on the fly' [13, 14]. Performance is analyzed in terms of both static and dynamic regrets, which establish conditions that the novel algorithm must satisfy in order to track nonlinear learning functions in environments with *unknown* dynamics. Tests with a number of real datasets corroborate the effectiveness of the novel algorithms [15–17].

1.2.3 Tensor-based network topology identification

Structural equation modeling (SEM) is a widely used tool for directed network topology inference. However, conventional SEMs require full knowledge of exogenous inputs, which may not be readily available in several practical settings [18, 19]. Prompted by this, Chapter 4 advocates a novel SEM-based topology inference approach that relies on factorizing a three-way tensor, constructed from the observed nodal data, using the tensor decomposition [20, 21]. Identifiability conditions are established to guarantee that the topology can be uniquely recovered. In addition, to facilitate real-time operation and inference of time-varying networks, an adaptive tensor decomposition scheme is developed to track the topology-revealing tensor factors. Extensive tests on real stock quote data demonstrate the effectiveness of the proposed tensor-based approach in identifying the causal dependencies among stocks even when stock prices for certain stocks are not fully available [20, 22, 23].

1.2.4 Nonlinear network topology identification

Chapter 5 further generalizes the SEMs and structural vector autoregressive models (SVARMs) to account for (possible) nonlinear interactions among nodes, which provides a powerful tool for identifying real-world networks [24–26]. The novel approach leverages kernels as a powerful encompassing framework for nonlinear modeling, and results in an efficient estimator with desirable complexity-expressibility tradeoffs. Pursuit of the novel kernel-based approach yields an estimator that promotes edge sparsity, a property exhibited by most real-world networks, such as brain and social networks. Experiments on a real gene expression dataset, as well as brain datasets illustrate that the novel method can unveil interesting new edges that were not revealed by conventional linear methods, which could shed more light on regulatory behavior of human genes, and brain dynamics at the seizure onset [27, 28].

1.2.5 Scalable nonlinear learning over graphs

With insights gained from previous works, Chapter 6 broadens the algorithmic and performance analysis framework to online scalable learning over possibly dynamic networks. Our results reveal the benefits of considering the underlying network topologies for classification and clustering [29, 30], as well as dimensionality reduction of data observed over graphs [31, 32]. [1]

Furthermore, real-world networks may have very large size, and nodal attributes can be unavailable to a number of nodes. Existing inference methods over graphs typically assume that the network size is fixed. However, new nodes can emerge over time, which necessitates real time analytics of growing network data [19]. In order to cope with newly-joining nodes, and to meet potential privacy constraints, Chapter 6 casts inference of nodal attributes as an online function learning task, and develops an adaptive *privacy-preserving* approach that is scalable to large-size dynamic networks. The novel method is further capable of effectively providing real-time evaluation of growing networks with newly-joining nodes without resorting to a batch solver. In addition, the novel scheme only relies on an *encrypted* version of each node's connectivity in order to learn the nodal attributes, which promotes privacy. Experiments on real email and citation network datasets corroborate the effectiveness of the proposed methods [33].

1.3 Notational conventions

Bold uppercase (lowercase) letters will denote matrices (column vectors), while operators $(\cdot)^\top$ and $\lambda_{\max}(\cdot)$, will stand for matrix transposition and maximum eigenvalue, respectively. The identity matrix will be denoted by $\mathbf{I}$, while ℓ_p, and Frobenius norms will be denoted by $\|.\|_p$ and $\|.\|_F$, respectively. The operator vec$(.)$ will vertically stack columns of its matrix argument, to form a vector. Finally, $\mathbf{A} \otimes \mathbf{B}$ will denote the Kronecker product of matrices $\mathbf{A}$ and $\mathbf{B}$, while $\mathbf{A} \odot \mathbf{B}$ will denote their Khatri-Rao product, namely, $\mathbf{A} \odot \mathbf{B} := [\mathbf{a}_1 \otimes \mathbf{b}_1, \dots \mathbf{a}_N \otimes \mathbf{b}_N]$, where $\mathbf{A} := [\mathbf{a}_1, \dots, \mathbf{a}_N]$ and $\mathbf{B} := [\mathbf{b}_1, \dots, \mathbf{b}_N]$. The projection operator $[\mathbf{a}]^+ := \max\{\mathbf{a}, \mathbf{0}\}$ is defined entrywise. Symbol $\dagger$ represents the Hermitian operator, while the indicator function $\mathbb{1}_{\{A\}}$ takes the value 1 when the event A holds, and 0 otherwise; $\mathbb{E}$ denotes the expectation, while $\langle \cdot, \cdot \rangle$ and $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ the vector inner product in Euclidean and Hilbert spaces, respectively.

[1]The corresponding publication was among the finalists for best paper award at the CAMSAP-2017 conference.

Chapter 2

Scalable subspace learning for big Streaming Categorical data

2.1 Introduction

Principal component analysis (PCA) is arguably the most popular tool for dimensionality reduction, with numerous applications in science and engineering [1]. It is however primarily designed to sketch high-dimensional data with analog-amplitude values, and does not suit categorical data emerging for instance, with recommender systems. Categorical PCA seeks a low-dimensional sketch of the high-dimensional categorical data to render affordable downstream machine learning tasks such as imputation, classification, and clustering; see e.g., [2–8]. However, the growing scale of nowadays ‘Big Data’ applications, such as recommender systems (e.g., NetFlix) with millions of users rating thousands of movies, pose extra challenges: (c1) the sheer volume of data approaches the computational and storage limits; (c2) new releases demand real-time processing for recommendations; and (c3) absent data entries, corresponding to missing user ratings.

Relation to prior work. Past works on categorical PCA focus on binary PCA, and rely on logistic-regression entailing (bi)linear models; see e.g., [3, 4, 6]. The work in [3] assumes that the log-odds matrix of the data lies in a linear low-dimensional subspace. The approach in [6] further imposes a Gaussian prior on the sketch, whereas the one in [4] promotes sparsity for the subspace to regularize the log-likelihood, which is then maximized using a batch majorization-minimization (MM) scheme. In a similar vein, binary matrix factorization has also been employed for dimensionality reduction when batch processing is affordable; see e.g., [34–37]. Other techniques for summarizing discrete-valued data include multidimensional scaling [38], and the k-modes algorithm [39], which extends k-means [40] to the discrete domain by adopting a proper dissimilarity measure. For streaming datasets, [8] proposes an online sketching scheme based on logistic PCA when all data entries are present, and an online binary dictionary learning algorithm has been developed in [11]. *All in all, the prior art is for the most developed for binary data, and either assumes the data have no missing entries, or, it relies on batch processing.*

Contributions. To cope with challenges (c1)-(c3), the present chapter brings forth a novel categorical subspace learning (CSL) scheme that unravels the latent structure behind the categorical data for three popular bilinear schemes; namely, Probit, Tobit, and Logit [41]. The Probit model treats categorical data as quantized values of a certain analog-amplitude vector that lies in a linear low-dimensional subspace. Tobit is the model of choice for censoring, while the probabilistic Logit model generalizes logistic regression to the unsupervised case. The bilinear models in this chapter can accommodate finite-alphabet datasets, and can also interpolate missing entries via rank regularization. To this end, the log-likelihood is regularized with a term corresponding to the rank of the underlying analog-valued data matrix. Leveraging a decomposable variant of the nuclear-norm, a recursive nonconvex program is then formulated, and solved online via stochastic alternating minimization. The resultant procedure alternates between sketching the new datum and refining the latent subspace via stochastic gradient descent to extract the information present in the new datum. This leads to lightweight first-order iterates that are nicely parallelizable across the latent subspace dimension, and thus implemented very efficiently via graphical processing units (GPUs).

The deterministic Probit and Tobit models adopt pre-determined quantization thresholds, which we further adjust to enhance the predictive power of the categorical models. To this end, the first-order iterates are modified to jointly learn the quantizer thresholds as well as the latent subspace. Performance of the subspace iterates is also analyzed for both *finite* and *infinite* data streams, where the former relies on martingale sequences to prove asymptotic convergence of the subspace to the stationary point set of the batch maximum likelihood (ML) estimator. For finite data streams, an unsupervised notion of regret is adopted to derive sublinear regret bounds for the empirical cost. Extensive simulated tests are performed with synthetic and real datasets for classification of chess-game scenaria, and interpolation of absent ratings in movie recommender systems. They corroborate the convergence and effectiveness of the novel sketching scheme in terms of accuracy and runtime relative to the existing alternatives.

To better place the present work in context, it is important to differentiate its novelties from online dictionary learning [42, 43], and online subspace learning [44], which is the closest to this contribution. To stand alone, the present work is motivated by *dimensionality reduction* of *categorical* data as a feature extraction scheme for *machine learning* tasks. In contrast, for *analog-valued data* [44] focuses on *interpolation* of misses, and [42, 43] deal with *denoising* when no misses are present. The algorithm and its asymptotic convergence analysis are inspired by [42, 43] and [44]. However, contrary to [42, 43] and [44], this work deals with categorical data; it offers regret analysis for finite data streams; and performs sketch evaluation with real-world datasets including "King-Rook versus King-Pawn" and "Movie-Lens" for chess-game classification and movie recommendation, respectively.

The rest of this chapter is organized as follows. Section 2.2 presents preliminaries, and states the problem. Section 2.3 formulates the ML estimator with rank regularization, based on which Section 2.4 develops subspace learning algorithms for online sketching via stochastic alternating minimization as well as learning the quantizer, while the performance of first-order subspace iterates is analyzed in Section 2.5. Section 2.6 reports the numerical tests with synthetic and real datasets, while conclusions are drawn in Section 2.7.

2.2 Preliminaries and Problem Statement

Consider the high-dimensional $D \times 1$ vectors $\{\mathbf{y}_\tau\}_{\tau=1}^{T}$ with categorical entries drawn from a J-element alphabet $\mathcal{S} := \{s_0, \ldots, s_{J-1}\}$. For instance, in movie recommender systems $\mathbf{y}_t$ represents the users' categorical ratings (e.g., "good" or "bad") for the t-th movie. Apparently, each user can only rate a small fraction of movies, and thus ratings for a sizable portion of movies may not be available. Let $\Omega_t \subseteq \{1, \ldots, D\}$ with cardinality $|\Omega_t|$ $(\ll D)$ denote the set of available entries (user ratings) associated with the t-th movie. With the *partial* categorical data $\{y_{t,i},\ i \in \Omega_t\}_{t=1}^{T} \in \mathcal{S}^{\mathcal{D}}$, categorical PCA seeks a low-dimensional (sketched) set of features $\{\boldsymbol{\psi}_\tau\}_{\tau=1}^{T} \in \mathbb{R}^d$ (with $d \ll D$), which render affordable downstream inference tasks such as regression, prediction, interpolation, classification, or, clustering; see e.g., [3–8]. Aiming at

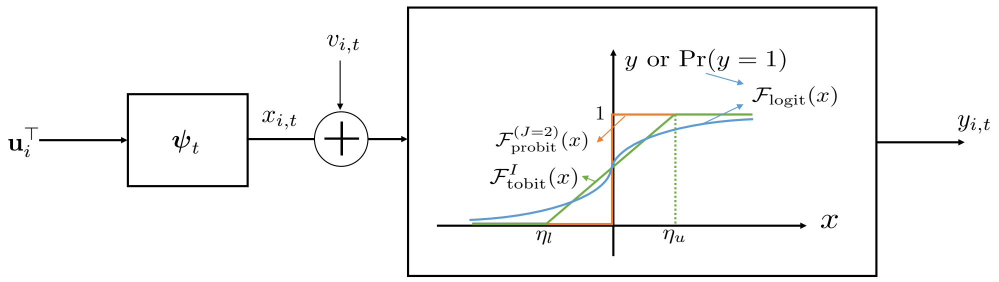

$v_{i,t}$
$\mathbf{u}_i^\top$
$\boldsymbol{\psi}_t$
$x_{i,t}$
y or $\Pr(y=1)$
$\mathcal{F}_{\text{logit}}(x)$
1
$\mathcal{F}^{(J=2)}_{\text{probit}}(x)$
$\mathcal{F}^{I}_{\text{tobit}}(x)$
x
η_l
η_u
$y_{i,t}$

Figure 2.1: Illustration of the considered models, namely Probit, Tobit and Logit.

a related objective, the present work builds on three *unsupervised* categorical models that are described next.

2.2.1 Blind Probit model

The Probit model regards $\mathcal{S}$ as the range space of a J-element quantization mapping

$$\mathcal{F}^{(J)}_{\text{probit}}(x) := s_j \ \text{ if } \ x \in (\eta_j, \eta_{j+1}] \qquad \text{for } j = 0, 1, \ldots, J-1 \tag{2.1}$$

with $\{\eta_j\}$ denoting known quantization thresholds. The categorical vectors $\{\mathbf{y}_t\}_{t=1}^{T}$ are then viewed as the quantized versions of certain analog-valued data vectors that belong (or lie close) to a linear low-dimensional subspace $\mathcal{U}$. Specifically, the i-th entry admits the following quantized bilinear model

$$y_{i,t} = \mathcal{F}^{(J)}_{\text{probit}}(x_{i,t} + v_{i,t}) \tag{2.2a}$$

$$x_{i,t} := \mathbf{u}_i^\top \boldsymbol{\psi}_t, \qquad i \in \Omega_t \tag{2.2b}$$

where $\boldsymbol{\psi}_t \in \mathbb{R}^d$ denotes the projection of $\mathbf{y}_t \in \mathbb{R}^D$ onto the low-dimensional ($d < D$) subspace $\mathcal{U}$; see also Fig. 2.1. Columns of the matrix $\mathbf{U} := [\mathbf{u}_1, \ldots, \mathbf{u}_D]^\top$, where $\mathbf{u}_i^\top \in \mathbb{R}^d$ denotes the i-th row of $\mathbf{U}$ span the linear subspace $\mathcal{U}$. The noise $v_{i,t}$ also accounts for errors and unmodeled dynamics.

Probit regression is widely used in practice for modeling categorical responses [45]. Consider for instance the survival outcome (alive, or, dead) for patients with a certain disease over a period of time. The patient's survival, or, death is a binary response that depends upon several factors such as age, weight, gender, as well as the treatment dose and specifications.

Our goal of finding $\{\boldsymbol{\psi}_t\}_{t=1}^T$ and $\mathbf{U}$ corresponds to blind regression given finite-alphabet $\{y_{i,t}\}$, while for $\mathbf{U}$ known, it is closely related to nonblind Probit-based classification.

2.2.2 Blind Tobit model

Acquired data in practice can be censored to e.g., lie in a prescribed range, for further processing. Given thresholds η_l and η_u, a typical censoring rule discards large data entries based on

$$\mathcal{F}_{\text{tobit}}^{I}(x) := \begin{cases} \eta_u & x \geq \eta_u \\ \eta_l & x \leq \eta_l \\ x & x \in (\eta_l, \eta_u). \end{cases} \tag{2.3}$$

Alternatively, one can think of a censoring rule that removes small data entries as effected by

$$\mathcal{F}_{\text{tobit}}^{\text{II}}(x) := \begin{cases} x & x \geq \eta_u \\ x & x \leq \eta_l \\ \eta & x \in (\eta_l, \eta_u). \end{cases} \tag{2.4}$$

As with the Probit model, to gain practical insight about the Tobit model, note that if the patient dies naturally within the study period, one knows precisely the survival time. However, if the patient dies before or after the study, where no accurate data is collected, only an upper or a lower bound is available on the patient age. Tobit models have been shown useful in big data applications for selecting informative observations [46].

Similar to (2.2), one can postulate the censored bilinear Tobit model (see also Fig. 2.1)

$$y_{i,t} = \mathcal{F}_{\text{tobit}}(x_{i,t} + v_{i,t}) \tag{2.5a}$$

$$x_{i,t} := \mathbf{u}_i^\top \boldsymbol{\psi}_t, \qquad i \in \Omega_t. \tag{2.5b}$$

2.2.3 Blind Logit model

Probit and Tobit adopt deterministic data-generating functions $\mathcal{F}$ and rely on nonlinear regression to predict missing categorical (hard) data. Inspired by logistic regression, Logit relies on a *probabilistic* (soft) model to predict label probabilities [47]. Suppose $\{y_{i,t}\}$ are mutually independent random variables, where the i-th entry $y_{i,t}$ is Bernoulli distributed with success probability $\pi_{i,t} := \Pr(y_{i,t} = 1)$. Define also the log-likelihood ratio $x_{i,t} := \log\{\pi_{i,t}/(1 - \pi_{i,t})\}$, which upon solving for $\pi_{i,t}$ yields the Logit function $\pi(x) := \{1 + \exp(x)\}^{-1}$.

The Logit model postulates that the log-likelihood ratio sequence $\{x_{i,t}\}$ belongs to a linear low-dimensional subspace spanned by the matrix $\mathbf{U}$; that is, $x_{i,t} := \mathbf{u}_i^\top \boldsymbol{\psi}_t$ for some $\boldsymbol{\psi}_t$, and for the binary case ($s \in \{0, 1\}$), the categorical data probability is thus expressed as

$$\begin{aligned} \mathcal{F}_{\text{logit}}(x_{i,t}) &:= \Pr(y_{i,t} = s) \\ &= \frac{1}{1 + \exp((1 - 2s)x_{i,t})}, \quad i \in \Omega_t. \end{aligned} \tag{2.6}$$

Likewise for the multibit Logit with each entry chosen from a J-element alphabet, $J-1$ bilinear Logit models start from the log-likelihood ratio

$$\log \frac{\Pr(y_{i,t} = s_j)}{\Pr(y_{i,t} = s_0)} = \boldsymbol{\psi}_t^\top \mathbf{u}_i^{(j)}, \quad j = 1, \ldots, J - 1 \tag{2.7}$$

where $\mathbf{u}_i^{(j)}$ is the predictor for the j-th class, and adopt the soft data model to arrive at (cf. (2.6))

$$\Pr(y_{i,t} = s_j) = \frac{\exp(\boldsymbol{\psi}_t^\top \mathbf{u}_i^{(j)})}{1 + \sum_{k=1}^{J-1} \exp(\boldsymbol{\psi}_t^\top \mathbf{u}_i^{(k)})}, \quad j = 1, \ldots, J - 1. \tag{2.8}$$

Different from (2.2) and (2.5) where (hard) categorical data $y_{i,t}$ are nonlinear functions of $x_{i,t}$, Logit deals with (soft) probability data $\Pr(y_{i,t} = s)$, expressed in (2.8) as a nonlinear function of $x_{i,t}$. With the patient's survival example in mind, the Logit model can predict the survival chance within a certain period of time [45].

Given $\{y_{i,t}\}$, the ensuing section will develop ML estimators of $\mathbf{U}$ and $\{\boldsymbol{\psi}_t\}$ for the three models introduced in this section, namely (2.2), (2.5), and (2.8).

2.3 Rank-Regularized ML Estimation

In what follows the likelihood function will be derived first, when the additive noise $v_{i,t} \sim \mathcal{N}(0, \sigma^2)$ is independent and identically distributed (i.i.d.), zero-mean Gaussian, with variance σ^2. As a result, available categorical entries $\{y_{i,t}\}$ are independent across i and t.

For the Probit model in (2.2), the per-categorical-entry likelihood can be written as

$$\Pr(y_{i,t};\mathbf{u}_i,\boldsymbol{\psi}_t)=\prod_{j=0}^{J-1}\Pr\{x_{i,t}\in(\eta_j,\eta_{j+1}]\}^{\mathcal{I}(y_{i,t}=s_j)}$$

$$=\prod_{j=0}^{J-1}\left[\mathrm{Q}\left(\frac{\eta_j-\mathbf{u}_i^\top\boldsymbol{\psi}_t}{\sigma}\right)-\mathrm{Q}\left(\frac{\eta_{j+1}-\mathbf{u}_i^\top\boldsymbol{\psi}_t}{\sigma}\right)\right]^{\mathcal{I}(y_{i,t}=s_j)} \tag{2.9}$$

where $\mathcal{I}(\epsilon)$ is the indicator function, and $\mathrm{Q}(\cdot)$ denotes the standard Gaussian tail function. Upon collecting the low-dimensional representations in a matrix $\boldsymbol{\Psi}:=[\boldsymbol{\psi}_1,\ldots,\boldsymbol{\psi}_T]$, the log-likelihood of the available categorical data can be expressed as

$$\log\mathcal{L}_{\text{probit}}\Big(\{y_{i,\tau},\ i\in\Omega_\tau\}_{\tau=1}^{T};\mathbf{U},\boldsymbol{\Psi}\Big)$$

$$=\ \sum_{\tau=1}^{T}\sum_{i\in\Omega_\tau}\log\ell_{\text{probit}}(y_{i,\tau};\mathbf{u}_i,\boldsymbol{\psi}_\tau) \tag{2.10a}$$

with

$$\log\ell_{\text{probit}}(y_{i,t};\mathbf{u}_i,\boldsymbol{\psi}_t):=\sum_{j=0}^{J-1}\mathcal{I}(y_{i,t}=s_j)$$

$$\times\log\left[\mathrm{Q}\left(\frac{\eta_j-\mathbf{u}_i^\top\boldsymbol{\psi}_t}{\sigma}\right)-\mathrm{Q}\left(\frac{\eta_{j+1}-\mathbf{u}_i^\top\boldsymbol{\psi}_t}{\sigma}\right)\right]\ . \tag{2.10b}$$

For the Tobit-I model in (2.3), one can readily derive the per-entry log-likelihood as

$$\ell_{\text{tobit}-\text{I}}(y_{i,t};\mathbf{u}_i,\boldsymbol{\psi}_t):=\phi\left(\frac{y_{i,t}-\mathbf{u}_i^\top\boldsymbol{\psi}_t}{\sigma}\right)\mathcal{I}(y_{i,t}\in(\eta_l,\eta_u))$$

$$+\mathrm{Q}\left(\frac{\eta_u-\mathbf{u}_i^\top\boldsymbol{\psi}_t}{\sigma}\right)\mathcal{I}(y_{i,t}=\eta_u)$$

$$+\Big[1-\mathrm{Q}\left(\frac{\eta_l-\mathbf{u}_i^\top\boldsymbol{\psi}_t}{\sigma}\right)\Big]\mathcal{I}(y_{i,t}=\eta_l) \tag{2.11a}$$

with $\phi(\cdot)$ denoting the probability density function (pdf) of the standardized Gaussian $\mathcal{N}(0,1)$.

Likewise, the corresponding log-likelihood for the Tobit-II in (2.4) can be represented as

$$\ell_{\text{tobit}-\text{II}}(y_{i,t};\mathbf{u}_i,\boldsymbol{\psi}_t)$$

$$
\begin{aligned}
&:= \left[\mathrm{Q}\left(\frac{\eta_j - \mathbf{u}_i^\top \boldsymbol{\psi}_t}{\sigma}\right) - \mathrm{Q}\left(\frac{\eta_{j+1} - \mathbf{u}_i^\top \boldsymbol{\psi}_t}{\sigma}\right)\right] \mathcal{I}(y_{i,t} = \eta) \\
&\quad + \phi\left(\frac{y_{i,t} - \mathbf{u}_i^\top \boldsymbol{\psi}_t}{\sigma}\right) \mathcal{I}(y_{i,t} \geq \eta_u) \\
&\quad + \phi\left(\frac{y_{i,t} - \mathbf{u}_i^\top \boldsymbol{\psi}_t}{\sigma}\right) \mathcal{I}(y_{i,t} \leq \eta_l). \qquad (2.11\text{b})
\end{aligned}
$$

The overall log-likelihood for censored data is then obtained similar to (2.10a).

Finally, for the Logit model, based on the per-datum likelihood in (2.8), the per-entry log likelihood can be written as

$$
\ell_{\text{logit}}(y_{i,t}; \mathbf{u}_i, \boldsymbol{\psi}_t) := \sum_{j=0}^{J-1} \mathcal{I}(y_{i,t} = s_j) \log \left[\frac{\exp(\boldsymbol{\psi}_t^\top \mathbf{u}_i^{(j)})}{1 + \sum_{k=0}^{J-1} \exp(\boldsymbol{\psi}_t^\top \mathbf{u}_i^{(k)})} \right] \qquad (2.12)
$$

and consequently the overall log-likelihood can be obtained by substituting (2.12) into the counterpart of (2.10a), where Probit is replaced by Logit.

So far (2.10), (2.11), and (2.12) provide the building blocks of our ML criterion for the Probit, Tobit, and Logit model, respectively. In our ML approach however, we have not yet accounted for the low-rank property inherent to our data $\{y_{i,t}\}$, or, their probabilities $\{\Pr(y_{i,t} = s_j)\}$. This is the subject dealt with in the next subsection.

Collect entries $x_{i,t} = \mathbf{u}_i^\top \boldsymbol{\psi}_t$ to form the $D \times 1$ vector $\mathbf{x}_t = \mathbf{U}\boldsymbol{\psi}_t$. Since the stream $\{\mathbf{x}_t\}$ lies in a linear low-dimensional subspace, $\mathbf{X} := [\mathbf{x}_1, \ldots, \mathbf{x}_T] = \mathbf{U}\boldsymbol{\Psi}$ is a low-rank matrix. A natural way to account for this property is to constrain the likelihood maximization over the set of low-rank matrices. However, since minimizing rank is in general NP-hard, the nuclear norm $\|\mathbf{X}\|_* := \sum_i \sigma_i(\mathbf{X})$ (where σ_i signifies the i-th singular value) will be adopted as a convex surrogate for the rank [48]. These considerations prompted us to minimize the regularized negative log-likelihood

$$
\text{(P1)} \qquad \min_{\mathbf{X}=\mathbf{U}\boldsymbol{\Psi}} -\log \mathcal{L}\Big(\{y_{i,\tau},\ i \in \Omega_\tau\}_{\tau=1}^T; \mathbf{U}, \boldsymbol{\Psi}\Big) + \lambda \|\mathbf{X}\|_*
$$

where $\mathcal{L}$ collectively refers to the likelihood for any of the models in (2.2), (2.5), or (2.7). The parameter λ also controls the dimension of the latent subspace, and it can be tuned using cross validation. For the binary case ($J = 2$), the nuclear-norm regularization in (P1) has been shown under mild conditions to offer reconstruction guarantees for the Probit and Logit models [49].

Apparently, the regularizer in (P1) entangles the data points, and as a result it challenges the development of efficient online solvers. To mitigate this computational challenge, the following bilinear characterization of the nuclear-norm is adopted (cf. [44, 50, 51])

$$\begin{aligned} \|\mathbf{X}\|_* = &\min_{\{\mathbf{U},\boldsymbol{\Psi}\}} \frac{1}{2}\left(\|\mathbf{U}\|_F^2 + \|\boldsymbol{\Psi}\|_F^2\right) \\ &\text{s. to} \quad \mathbf{X} = \mathbf{U}\boldsymbol{\Psi} \end{aligned} \tag{2.13}$$

where the minimization is over all possible bilinear factorizations of $\mathbf{X}$. Bypassing the need for calculating singular values of $\mathbf{X}$ whose size grows with time, this characterization of the nuclear norm not only effects a surrogate of the rank constraint, but also decouples variables across time, thus facilitating online optimization tasks [44, 50]. Utilizing (2.13) into (P1) after dropping the min operation, yields

$$\text{(P2)} \qquad \min_{\{\mathbf{U},\boldsymbol{\Psi}\}} -\log \mathcal{L}\Big(\{y_{i,\tau},\ i \in \Omega_\tau\}_{\tau=1}^{T}; \mathbf{U}, \boldsymbol{\Psi}\Big) + \frac{\lambda}{2}\left(\|\mathbf{U}\|_F^2 + \|\boldsymbol{\Psi}\|_F^2\right).$$

Since the min operation is in effect at the optimum, it can be easily seen that the solutions of (P2) and (P1) coincide [50]. For a moderate number of data entries D and instants T, if the entire data is available in batch, one can develop alternating minimization algorithms along the lines of [50]. This amounts to cycling over two groups of variables, namely $\{\mathbf{U}, \boldsymbol{\Psi}\}$, to jointly refine the sketch $\boldsymbol{\Psi}$ and the subspace $\mathbf{U}$. However, for 'Big Data' applications with (large D) streaming over time ($T \to \infty$), the size of $\boldsymbol{\Psi}$ grows; thus, batch solvers become prohibitively complex, which well motivates the recursive solvers of the ensuing section.

2.4 Online Categorical Subspace Learning

With modern 'Big Data' applications, the massive amount of available data makes it impractical to store and process the data in an offline fashion. Furthermore, in many settings, the data are acquired sequentially over time and there is a need for real-time processing. In either case, practical limitations call for online schemes, capable of refining the sketch by adjusting the learned subspace to each new datum 'on the fly.' With this in mind, we recast (P2) to minimize the following empirical cost

$$\text{(P3)} \qquad \min_{\{\boldsymbol{\psi}_\tau\}_{\tau=1}^t, \mathbf{U}} \quad \frac{1}{t}\sum_{\tau=1}^{t} g_\tau\Big(\{y_{i,\tau}\}_{i\in\Omega_\tau}; \boldsymbol{\psi}_\tau, \mathbf{U}\Big)$$

where the instantaneous cost g_τ corresponding to the τ-th datum is given by

$$\begin{aligned} & g_\tau\big(\{y_{i,\tau}\}_{i\in\Omega_\tau}; \boldsymbol{\psi}_\tau, \mathbf{U}\big) \\ & := -\sum_{i\in\Omega_\tau} \log \ell(y_{i,\tau}; \boldsymbol{\psi}_\tau, \mathbf{u}_i) + \frac{\lambda}{2t}\sum_{i=1}^{D} \|\mathbf{u}_i\|_2^2 + \frac{\lambda}{2}\|\boldsymbol{\psi}_\tau\|_2^2. \end{aligned} \tag{2.14}$$

It is important to recognize that different from our schemes in [50] and [44], which rely on analog-valued data, the nonlinear cost in (P3) entails categorical data and Gaussian tail functions that challenge algorithmic derivations. This is further elaborated next.

2.4.1 First-order alternating minimization algorithms

To effectively solve (P3) for streaming data, an iterative alternating minimization (AM) method is adopted, where the iteration index coincides with the acquisition time. The sought AM scheme comprises two learning steps. Upon acquiring $\{y_{i,t}\}_{i\in\Omega_t}$ at time instant t, the first step (S1) embeds the data into the latent low-dimensional subspace, updates the features $\boldsymbol{\psi}_t$, and as a byproduct imputes the missing data entries. Subsequently, step (S2) refines the latent subspace according to the latest imputed datum.

In (S1), given the subspace at the previous update $\mathbf{U}[t-1]$, the embedding is obtained as

$$\boldsymbol{\psi}_t = \arg\min_{\boldsymbol{\psi}\in\mathbb{R}^d} g_t\big(\{y_{i,t}\}_{i\in\Omega_t}; \boldsymbol{\psi}, \mathbf{U}[t-1]\big). \tag{2.15}$$

This amounts to a nonlinear ridge-regression task, given categorical $\{y_{i,t}\}_{i\in\Omega_t}$ with misses, along with their predictors $\{\mathbf{u}_i[t-1]\}_{i\in\Omega_t}$ corresponding to the rows of $\mathbf{U}[t-1]$. In the binary Probit model, the embedding $\boldsymbol{\psi}_t$ can also be viewed as the classifying hyperplane that assigns vectors $\mathbf{u}_i[t-1],\ i\in\Omega_t$, to their labels. With this interpretation, the j-th absent entry can be imputed by projecting $\mathbf{u}_j[t-1]$ onto the hyperplane $\boldsymbol{\psi}_t$ that is then quantized to return the label sign($\mathbf{u}_j^\top[t-1]\boldsymbol{\psi}_t$). Similarly, if the Logit model is adopted, (2.15) can be viewed as training a binary logistic regression classifier.

Algorithm 1 Online rank-regularized ML sketching for the Probit model

```
input: $\{y_{i,\tau},\ i \in \Omega_\tau\}_{\tau=1}^{T}$, $\{\mu_t\}$, $\lambda$
initialize $\mathbf{U}[0]$ at random.
for $t = 1, 2, \ldots$ do
    (S1) Sketching via first-order Algorithm 1a or
    second-order Algorithm 1b
    $\boldsymbol{\psi}_t = \arg\min_{\boldsymbol{\psi}\in\mathbb{R}^d} g_t\big(\{y_{i,t}\}_{i\in\Omega_t}; \boldsymbol{\psi}, \mathbf{U}[t-1]\big)$
    (S2) Parallel subspace refinement ($i \in \{1,\ldots,D\}$)
    $z^i_{j,t-1} := \sigma^{-1}(\eta_j - \mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t)$
    $f_{i,t} := \sum_{j=0}^{J-1}\mathcal{I}(y_{i,t}=s_j)\Big[\phi(z^i_{j,t-1}) - \phi(z^i_{j+1,t-1})\Big]$
    $w_{i,t} := \sum_{j=0}^{J-1}\mathcal{I}(y_{i,t}=s_j)\Big[\mathrm{Q}(z^i_{j,t-1}) - \mathrm{Q}(z^i_{j+1,t-1})\Big]$
    $\mathbf{u}_i[t] = \begin{cases}(1-\lambda\mu_t/t)\mathbf{u}_i[t-1] + \mu_t(f_{i,t}/w_{i,t})\boldsymbol{\psi}_t, & i\in\Omega_t\\ (1-\lambda\mu_t/t)\mathbf{u}_i[t-1], & i\notin\Omega_t\end{cases}$
end for
return $\Big(\mathbf{U}[t], \{\boldsymbol{\psi}_\tau\}_{\tau=1}^{t}\Big)$
```

The optimization problem (2.15) involves only $d \ll D$ variables, and can be readily solved using off-the-shelf solvers, such as gradient descent or Newton method. The recursions for the Probit model are derived after regularizing (2.10) as in (2.14), and the corresponding iterates are listed in Algorithm 1.

With the sketch $\{\boldsymbol{\psi}_\tau\}_{\tau=1}^{t}$ at hand, (S2) proceeds to update the subspace in (P3). This is however a daunting task since for the considered categorical models the regularized loss g_t relates to the latent subspace $\mathbf{U}$ in a complicated way (through functions of the Gaussian pdf for the Probit and Tobit, and exponential functions for the Logit model), which precludes closed-form solutions. To bypass this computational hurdle, we will adopt an inexact solution of (P3). The basic idea leverages the empirical cost of (P3) to incorporate the information of the latest datum through a stochastic gradient descent iteration. In essence, at iteration (time) t the old subspace estimate is updated by moving (with an appropriate step size) along the opposite gradient direction of g_t incurred by the latest datum. All in all, this yields the recursion

$$\mathbf{u}_i[t] = \mathbf{u}_i[t-1] - \mu_t \nabla_{\mathbf{u}_i} g_t\big(\{y_{i,t}\}_{i\in\Omega_t}; \boldsymbol{\psi}_t, \mathbf{U}[t-1]\big) \tag{2.16}$$

Algorithm 1a Gradient-descent algorithm to obtain the sketch for the Probit model

```
input: $\{y_{i,t},\ i \in \Omega_t\}$, $\{\eta_j\}$, $\{\beta_k\}$, $\lambda$, $K$, $\sigma$, $\mathbf{U}[t-1]$
initialize: $\boldsymbol{\psi}_t^{(0)}$
for $k = 1, \ldots, K$ do
    $a^i_{j,t-1} = \sigma^{-1}(\eta_j - \mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t^{(k-1)})$
    $\epsilon_i := \sum_{j=0}^{J-1} \mathcal{I}(y_{i,t} = s_j) \left[\frac{\phi(a^i_{j,t}) - \phi(a^i_{j+1,t})}{\mathrm{Q}(a^i_{j,t-1}) - \mathrm{Q}(a^i_{j+1,t-1})}\right]$
    $\boldsymbol{\psi}_t^{(k)} = (1-\beta_k)\boldsymbol{\psi}_t^{(k-1)} + \beta_k \sum_{i\in\Omega_t} \epsilon_i \mathbf{u}_i[t-1]$
end for
return $\boldsymbol{\psi}_t^{(K)}$
```

where μ_t is the step size that can vary across time.

For the Probit model, the gradient is simply obtained as

$$\nabla_{\mathbf{u}_i} g_t^{(\mathrm{probit})}\big(\{y_{i,t}\}_{i\in\Omega_t}; \boldsymbol{\psi}_t, \mathbf{U}[t-1]\big) = -\frac{f(\mathbf{u}_i[t-1], \boldsymbol{\psi}_t)}{w(\mathbf{u}_i[t-1], \boldsymbol{\psi}_t)}\boldsymbol{\psi}_t + \frac{\lambda}{t}\mathbf{u}_i[t-1] \tag{2.17}$$

where the scalar functions f and w are given by

$$f(\mathbf{u}_i[t-1], \boldsymbol{\psi}_t) := \sum_{j=0}^{J-1} \mathcal{I}(y_{i,t} = s_j)\left[\phi(z^i_{j,t-1}) - \phi(z^i_{j+1,t-1})\right]$$

with $z^i_{j,t-1} := \sigma^{-1}(\eta_j - \mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t)$, and

$$w(\mathbf{u}_i[t-1], \boldsymbol{\psi}_t) := \sum_{j=0}^{J-1} \mathcal{I}(y_{i,t} = s_j)\left[\mathrm{Q}\left(z^i_{j,t-1}\right) - \mathrm{Q}(z^i_{j+1,t-1})\right].$$

For the Tobit-I model, the gradient is expressed as

$$\nabla_{\mathbf{u}_i} g_t^{(\mathrm{tobit-I})}\big(\{y_{i,t}\}_{i\in\Omega_t}; \boldsymbol{\psi}_t, \mathbf{U}[t-1]\big) = \begin{cases} -\frac{\left(y_{i,t} - \mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t\right)}{\sigma^2}\boldsymbol{\psi}_t + \frac{\lambda}{t}\mathbf{u}_i[t-1], & y_{i,t} \in (\eta_l, \eta_u) \\ \frac{\phi(z^i_{u,t})}{\sigma \mathrm{Q}(z^i_{u,t})}\boldsymbol{\psi}_t + \frac{\lambda}{t}\mathbf{u}_i[t-1], & y_{i,t} = \eta_u \\ \frac{\phi(z^i_{l,t})}{\sigma \mathrm{Q}(z^i_{l,t})}\boldsymbol{\psi}_t + \frac{\lambda}{t}\mathbf{u}_i[t-1], & y_{i,t} = \eta_l \end{cases} \tag{2.18}$$

Algorithm 1b Newton method to obtain the sketch for the Probit model

```
**input:** $\{y_{i,t},\ i \in \Omega_t\}$, $\{\eta_j\}$, $\{\beta_k\}$, $\lambda$, $K$, $\sigma$, $\mathbf{U}[t-1]$
**initialize:** $\boldsymbol{\psi}_t^{(0)}$
**for** $k = 1, 2, \ldots, K$ **do**
    $a_{j,t-1}^i := \sigma^{-1}(\eta_j - \mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t^{(k-1)})$
    $\theta_{i,t} := \sum_{j=0}^{J-1} \mathcal{I}(y_{i,t} = s_j) \times \left[a_{j,t-1}^i \phi(a_{j,t-1}^i) - a_{j+1,t-1}^i \phi(a_{j+1,t-1}^i)\right]$
    $f_{i,t} := \sum_{j=0}^{J-1} \mathcal{I}(y_{i,t} = s_j) \left[\phi(a_{j,t-1}^i) - \phi(a_{j+1,t-1}^i)\right]$
    $w_{i,t} := \sum_{j=0}^{J-1} \mathcal{I}(y_{i,t} = s_j) \left[\mathrm{Q}(a_{j,t-1}^i) - \mathrm{Q}(a_{j+1,t-1}^i)\right]$
    $\nabla_{\boldsymbol{\psi}_t} g_t = -\sum_{i \in \Omega_t} \frac{f_{i,t}}{w_{i,t}} \mathbf{u}_i[t-1] + \lambda \boldsymbol{\psi}_t^{(k-1)}$
    $\nabla^2_{\boldsymbol{\psi}_t} g_t = -\sum_{i \in \Omega_t} \left[\frac{f_{i,t}^2}{w_{i,t}^2} - \frac{\theta_{i,t}}{w_{i,t}}\right] \mathbf{u}_i[t-1]\mathbf{u}_i^\top[t-1] + \lambda \mathbf{I}$
    $\boldsymbol{\psi}_t^{(k)} = \boldsymbol{\psi}_t^{(k-1)} - \beta_k (\nabla^2_{\boldsymbol{\psi}_t} g_t)^{-1} \nabla_{\boldsymbol{\psi}_t} g_t$
**end for**
**return** $\boldsymbol{\psi}_t^{(K)}$
```

where $z_{u,t-1}^i := \sigma^{-1}\left(\eta_u - \mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t\right)$, and likewise for $z_{l,t-1}$. For the Tobit-II model, we have

$$\nabla_{\mathbf{u}_i} g_t^{(\text{tobit}-\text{II})}\left(\{y_{i,t}\}_{i \in \Omega_t}; \boldsymbol{\psi}_t, \mathbf{U}[t-1]\right)$$
$$= \begin{cases} -\frac{\phi(z_{l,t-1}^i) - \phi(z_{u,t-1}^i)}{\mathrm{Q}(z_{l,t-1}^i) - \mathrm{Q}(z_{u,t-1}^i)} \boldsymbol{\psi}_t + \frac{\lambda}{t}\mathbf{u}_i[t-1], & y_{i,t} \in (\eta_l, \eta_u) \\ -\frac{\left(y_{i,t} - \mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t\right)}{\sigma^2} \boldsymbol{\psi}_t + \frac{\lambda}{t}\mathbf{u}_i[t-1], & y_{i,t} = \eta_l, \text{or } \eta_u. \end{cases} \tag{2.19}$$

Finally, one can arrive at the gradient of the binary Logit model that is given by

$$\nabla_{\mathbf{u}_i} g_t^{(\text{logit})}\left(\{y_{i,t}\}_{i \in \Omega_t}; \boldsymbol{\psi}_t, \mathbf{U}[t-1]\right)$$
$$= \frac{(2y_{i,t} - 1)\exp\{(2y_{i,t} - 1)\mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t\}}{1 + \exp\{(2y_{i,t} - 1)\mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t\}} \boldsymbol{\psi}_t + \frac{\lambda}{t}\mathbf{u}_i[t-1]. \tag{2.20}$$

The subspace update (2.16) amounts to exactly solving a first-order approximation of the cost in (P3). The overall procedure is summarized in Algorithm 1 only for the Probit model, but it also applies for the Tobit and Logit models with obvious modifications for the gradient correction terms.

Remark 1 (Computational cost): The subspace update in Algorithm 1 is parallelizable across rows of $\mathbf{U}$ (D), and can be efficiently implemented on GPUs. The major complexity emanates from running the iterative Algorithm 1a or 1b, for obtaining $\boldsymbol{\psi}_t$. Fixing the maximum number of inner iterations to K, this demands $\mathcal{O}(Kd^2D)$ operations for Algorithm 1b, and $\mathcal{O}(KdD)$ operations for Algorithm 1a. Our empirical observations suggest that even an inexact solution of (S1) obtained by running Algorithm 1b with a few iterations K suffices for Algorithm 1 to converge. The remaining operations entail multiplications and additions of order $\mathcal{O}(D)$. The overall cost of the Algorithm 1 per iteration is $\mathcal{O}(Kd^2D)$, which is affordable since d is generally small.

2.4.2 Learning the quantizer

The Probit model discussed in the previous sections requires quantization thresholds $\{\eta_j\}_{j=0}^{J-1}$ to be available. These thresholds however add degrees of freedom, which can enhance the predictive power of the Probit based approach to modeling categorical data. While one can derive the general multibit case, to simplify exposition, consider the binary case with a single threshold η that is assumed fixed over time. With this in mind, (2.2) boils down to

$$y_{i,t} = \text{sign}(\mathbf{u}_i^\top \boldsymbol{\psi}_t + v_{i,t} - \eta). \tag{2.21}$$

To sketch big categorical data obeying (2.21), both $\mathbf{u}_i$ and η must be selected jointly. An estimate of these parameters can be found by jointly maximizing the rank-regularized likelihood in (P2), where the per-entry log-likelihood is now replaced by

$$\log \ell_{\text{probit}}(y_{i,t}; \mathbf{u}_i, \boldsymbol{\psi}_t, \eta) = \frac{1+y_{i,t}}{2} \log \mathrm{Q}\left(\frac{\eta - \mathbf{u}_i^\top \boldsymbol{\psi}_t}{\sigma}\right) + \frac{1-y_{i,t}}{2} \log \mathrm{Q}\left(\frac{\mathbf{u}_i^\top \boldsymbol{\psi}_t - \eta}{\sigma}\right) .$$

Accordingly, the updates for $\{\mathbf{u}_i\}$ and η are obtained by applying stochastic gradient descent to the empirical loss in (P3).

The sketch and subspace updates are similar to (2.15) and (2.16), while η is updated as

$$\eta[t] = \eta[t-1] - \gamma_t \nabla_\eta g_t\big(\{y_{i,t}\}_{i\in\Omega_t}; \boldsymbol{\psi}_t, \mathbf{U}[t], \eta\big) \tag{2.22}$$

where the gradient with respect to η is readily expressed as

$$\nabla_{\eta} g_t = -\sum_{i\in\Omega_t} \zeta_{i,t-1} \tag{2.23}$$

where $\zeta_{i,t-1} := -b_{i,t-1}\sigma^{-1}\phi(b_{i,t-1})/\mathrm{Q}\left(b_{i,t-1}\right)$, and $b_{i,t-1} := \sigma^{-1}y_{i,t}\left(\eta[t-1] - \mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t\right)$.

Albeit more complex, analogous updates are possible for the multibit Probit, and likewise for designing the quantizer when Tobit and Logit models are adopted.

2.5 Performance Analysis

This section establishes convergence of the first-order iterates in Algorithm 1 for the considered categorical models, namely Probit, Tobit, and Logit. Both asymptotic and non-asymptotic analyses for infinite and finite data streams are considered. The asymptotic analysis relies heavily on quasi-martingale sequences [42], while for non-asymptotic analysis we draw from regret metric advances in online learning [8, 52–54].

2.5.1 Asymptotic convergence analysis

For *infinite* data streams, convergence analysis of our categorical subspace learning schemes is inspired by [42], and our precursors in [44] and [50]. In order to render analysis tractable, the following assumptions are adopted.

(as1) The data streams $\{\mathbf{y}_t\}_{t=1}^{\infty}$ and sampling patterns $\{\Omega_t\}_{t=1}^{\infty}$ form an i.i.d. process; and

(as2) the subspace sequence $\{\mathbf{U}[t]\}$ lies in a compact set.

To begin, rewrite the rank-regularized empirical cost in (P3) as

$$\min_{\mathbf{U}\in\mathbb{R}^{D\times d}} \quad C_t(\mathbf{U}) := \frac{1}{t}\sum_{\tau=1}^{t} g_\tau(\boldsymbol{\psi}_\tau, \mathbf{U}). \tag{2.24}$$

As argued earlier in Section 2.4, minimization of (2.24) becomes increasingly complex computationally as t grows. The subspace $\mathbf{U}[t]$ is estimated by the stochastic gradient-descent (SGD) iteration with an appropriate step size. Define the *approximate* cost

$$\check{C}_t(\mathbf{U}) = \frac{1}{t}\sum_{\tau=1}^{t} \check{g}_\tau(\boldsymbol{\psi}_\tau, \mathbf{U}) \tag{2.25}$$

where $\breve{g}_t$ is a quadratic upperbound for $g_t(\cdot)$ based on the second-order Taylor approximation around the latest subspace estimate $\mathbf{U}[t-1]$; that is

$$\breve{g}_t(\boldsymbol{\psi}_t, \mathbf{U}) = g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1]) + \langle \nabla_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1]), \mathbf{U} - \mathbf{U}[t-1] \rangle \tag{2.26}$$

$$+\frac{\alpha_t}{2} \|\mathbf{U} - \mathbf{U}[t-1]\|_F^2 \tag{2.27}$$

with $\alpha_t \geq \|\nabla^2_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])\|$. It is useful to recognize that the quadratic surrogate $\breve{g}_t(\cdot)$ is a tight approximation for g_t, since (i) it is an upperbound, i.e., $\breve{g}_t(\boldsymbol{\psi}_t, \mathbf{U}) \geq g_t(\boldsymbol{\psi}_t, \mathbf{U}),\ \forall \mathbf{U}$; and (ii), it is locally tight, i.e., $\breve{g}_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1]) = g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])$, with (iii) locally tight gradient, i.e., $\nabla \breve{g}_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1]) = \nabla g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])$. Taking the gradient of $\breve{C}_t$ w.r.t. $\mathbf{U}$, and after simple rearrangements as elaborated in [44], SGD iterations can be seen as minimizing the approximate cost (2.25). Furthermore, g_t is smooth as asserted next.

Lemma 1: *Under (as2), upon defining $\delta_1 := \Delta/\sigma^2$, $\delta_2 := (\Delta^2/\sigma^2 + 1)/\sigma^2$, and $\Delta := \eta_{J-1} - \eta_0$, for the gradient and Hessian of the per-entry loss for the Probit model, it holds that*

$$\|\nabla_{\mathbf{u}_i} g_t^{(\text{probit})}(\boldsymbol{\psi}_t, \mathbf{U})\|_2 \leq \delta_1 \|\boldsymbol{\psi}_t\|_2 + \frac{\lambda}{t} \|\mathbf{u}_i\|_2 \tag{2.28}$$

$$\|\nabla^2_{\mathbf{u}_i} g_t^{(\text{probit})}(\boldsymbol{\psi}_t, \mathbf{U})\| \leq \delta_2 \|\boldsymbol{\psi}_t\|_2^2 + \frac{\lambda}{t} \tag{2.29}$$

and consequently the per-entry cost $g_t^{(\text{probit})}(\boldsymbol{\psi}_t, \mathbf{U})$, and $\nabla g_t^{(\text{probit})}(\boldsymbol{\psi}_t, \mathbf{U})$ are Lipschitz continuous.

Proof: See the Appendix .

The convergence of subspace iterates can then be established following the machinery developed in [42]. In the sequel, technical details are skipped due to space limitations, but they follow arguments similar to those in [44]. The proof sketch entails the following two main steps.

(Step1) The approximate cost $\breve{C}_t(\mathbf{U}[t])$ asymptotically converges to $C_t(\mathbf{U}[t])$, which is, $\lim_{t\to\infty} |C_t(\mathbf{U}[t]) - \breve{C}_t(\mathbf{U}[t])| = 0$. The convergence follows the quasi-martingale property of $\{\breve{C}_t\}$ in the almost sure (a.s.) sense owing to the tightness of the surrogate function $\breve{g}_t$.

(Step2) Due to the regularity of g_t, asymptotic convergence of $\{C_t(\mathbf{U}[t]) - \breve{C}_t(\mathbf{U}[t])\} \to 0$ implies convergence of the associated gradient sequence, namely $\{\nabla C_t(\mathbf{U}[t]) - \nabla \breve{C}_t(\mathbf{U}[t])\} \to 0$, which ultimately leads to $\nabla C_t(\mathbf{U}[t]) \to \mathbf{0}$.

The projection coefficients $\boldsymbol{\psi}_t$ can be solved exactly using Newton iterations due to the convexity of $g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])$, when the subspace is frozen at $\mathbf{U}[t-1]$. This is formalized in the next lemma.

Lemma 2: *Under the Probit, Tobit-II, and Logit models, the per-entry regualrized-loss $g_t(\boldsymbol{\psi}, \mathbf{U})$ is bi-convex for the block variables $\boldsymbol{\psi}$ and $\mathbf{u}_i$.*

Proof: See the Appendix 2.8.2.

All in all, combining the previous arguments with Lemmas 1 and 2, the asymptotic convergence claim for the iterations of Algorithm 1 can be asserted as follows.

Proposition 1: *Suppose (as1)-(as2) hold, and choose the step-size sequence $\{\mu_t = 1/\bar{\alpha}_t\}$ where $\bar{\alpha}_t \geq ct$, and $\delta_2\|\boldsymbol{\psi}_t\|^2 + \lambda/t \leq \alpha_t \leq c'$ for constants $c, c' > 0$, and δ_2 as in Lemma 1. Then, the subspace sequence $\{\mathbf{U}[t]\}$ satisfies $\lim_{t\to\infty} \nabla_{\mathbf{U}} C_t(\mathbf{U}[t]) = \mathbf{0}$, which means that the subspace iterates asymptotically converge to the stationary-point set of the batch ML estimator (P1).*

Remark 2: Independence under (as1) is customary for tractability of analysis when studying the performance of online (adaptive) algorithms. Still, in accordance with the adaptive filtering folklore (see e.g., [55, p. 109]) the upshot of the analysis based on i.i.d. data extends accurately to the pragmatic setting whereby the data and missing patterns exhibit temporal correlations. Furthermore, compactness under (as2) can be ensured by imposing a norm constraint, namely $\|\mathbf{U}\|_F \leq B$, which simply normalizes the updated subspace per iteration of the stochastic gradient descent.

2.5.2 Regret analysis

For *finite* data streams, we will rely on the unsupervised formulation of regret analysis to assess the performance of online iterates, in terms of interpolating misses and denoising the available categorical data. Regret analysis was originally introduced for the online supervised learning scenario [53], where the ground-truth label is revealed after prediction to incur a loss whose gradient is used to guide the learning. In the considered unsupervised sketching task however, the true labels are not revealed, which challenges regret analysis. Unsupervised variations of regret have been lately introduced to deal with online dictionary learning [52], and sequential logistic PCA [8].

Prompted by the alternating nature of iterations, we adopt a variant of the unsupervised regret to assess the goodness of online subspace estimates in representing the partially available data. Specifically, at iteration t, we use the previous update $\mathbf{U}[t-1]$ to span the recent partial data, namely, $y_{i,t},\ i \in \Omega_t$. With $g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])$ being the loss incurred by the estimate $\mathbf{U}[t-1]$ for predicting the t-th datum, the cumulative online loss for a stream of size T is given by

$$\bar{C}_T := \frac{1}{T}\sum_{\tau=1}^{T} g_\tau(\boldsymbol{\psi}_\tau, \mathbf{U}[\tau-1]). \tag{2.30}$$

Further, we will assess the cost of the last estimate $\mathbf{U}[T]$ using

$$\hat{C}_T = \frac{1}{T}\sum_{\tau=1}^{T} g_\tau(\boldsymbol{\psi}_\tau, \mathbf{U}[T]). \tag{2.31}$$

Comparing the losses in (2.25), (2.30), and (2.31), with $C_T := \min_{\mathbf{U}} C_T(\mathbf{U})$, it clearly holds that

$$\check{C}_T \geq \hat{C}_T \geq \bar{C}_T \geq C_T. \tag{2.32}$$

Accordingly, for the sequence $\{\mathbf{U}[t]\}_{t=1}^{T}$, define the online regret

$$\mathcal{R}_T := \hat{C}_T - \bar{C}_T. \tag{2.33}$$

Our next goal is to investigate the convergence rate of the sequence $\{\mathcal{R}_T\}$ to zero as T grows. This is important particularly because it is known from Proposition 1 that $|\check{C}_t - C_t| \to 0$ as $t \to \infty$, and as a result $|\bar{C}_t - C_t| \to 0$ (cf. (2.32)). Due to the nonconvexity of the online subspace iterates, it is challenging to directly analyze how fast the online cumulative loss $\bar{C}_t$ approaches the optimal batch cost C_t. Instead, we will investigate whether $\hat{C}_t$ converges to $\bar{C}_t$.

In the sequel, to derive regret bounds we focus on the Probit model. However, the same analysis caries over to develop regret bounds for the Tobit and Logit models too.

Proposition 2: *If $\{\mathbf{U}[t]\}$ and $\{\boldsymbol{\psi}_t\}$ are uniformly bounded, i.e., $\|\mathbf{U}[t]\|_F \leq B_u$, and $\|\boldsymbol{\psi}_t\|_2 \leq B_\psi$ for constants $B_u, B_\psi > 0$, choosing a constant step size $\mu_t = \mu$, leads to a bounded regret as*

$$\mathcal{R}_T \leq \frac{B^2(\ln(T)+1)^2}{2\mu T} + \frac{5B^2}{6\mu T}$$

where $B := (\lambda B_u + \delta_1 B_\psi)/\rho$ is a constant not dependent of T, δ_1 as in Lemma 1, and ρ

denotes the strong convexity constant on $\bar{C}_T$.

Remark 3(Subpace Projection): Instead of assuming bounded subspace iterates, namely $\|\mathbf{U}[t]\|_F \leq B_u$, one can alternatively introduce an additional projection onto the B_u-ball given by $\{\mathbf{U}| \ \|\mathbf{U}\|_F \leq B_u\}$. This additional projection does not alter the asymptotic convergence result in Proposition 2 due to the non-expansiveness of the projection operator.

To place Proposition 2 in context, relevant regret analyses have been carried out for the dictionary learning [52], and the sequential logistic PCA [8]. Different from our scheme, [52] deals with overcomplete dictionary updates with sparsity-regularized projection coefficients, and assumes that the estimation error is uniformly bounded. The regret bound obtained in [8] for logistic PCA also assumes no absent data entires, and it is relatively loose since the regret does not vanish as $T \to \infty$.

The proof technique of Proposition 2 relies on the following lemma, which asserts that the distance between successive subspace estimates vanishes as fast as $o(1/t)$, a property that will be instrumental to establish sub-linearity of the regret later.

Lemma 3: *[50] Under (as2), it holds that*

$$\|\mathbf{U}[t] - \mathbf{U}[t-1]\|_F \leq \frac{B}{t}$$

for some constant $B := (\lambda B_u + \delta_1 B_\psi)/\rho$, where ρ denotes the strong convexity constant of $\bar{C}_t$.

Proof: See the Appendix 2.8.3.

Toward bounding the regret, consider the difference of the iterates (cf. (2.16))

$$\mathbf{U}[t] - \mathbf{U}[t-1] = -\mu_t \nabla_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1]). \tag{2.34}$$

Taking the Frobenius norm on both sides yields

$$\|\mathbf{U}[t] - \mathbf{U}t - 1\|_F = \mu_t \|\nabla_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])\|_F \tag{2.35}$$

and after appealing to Lemma 3, we arrive at

$$\|\nabla_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])\|_F \leq \frac{B}{\mu_t t}. \tag{2.36}$$

On the other hand, it is easy to verify that (cf. (2.35))

$$\begin{aligned}
&\|\mathbf{U}[t] - \mathbf{U}[T]\|_F^2 \\
&= \|\mathbf{U}[t-1] - \mathbf{U}[T] + \mathbf{U}[t] - \mathbf{U}[t-1]\|_F^2 \\
&= \|\mathbf{U}[t-1] - \mathbf{U}[T]\|_F^2 + \mu_t^2 \|\nabla_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])\|_F^2 \\
&\quad - 2\mu_t \langle \mathbf{U}[t] - \mathbf{U}[T], \nabla_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])\rangle
\end{aligned}$$

which after re-arranging yields

$$\begin{aligned}
&\langle \mathbf{U}[t] - \mathbf{U}[T], \nabla_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])\rangle \\
&\quad = \frac{\|\mathbf{U}[t-1] - \mathbf{U}[T]\|_F^2}{2\mu_t} + \frac{\mu_t \|\nabla_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])\|_F^2}{2} - \frac{\|\mathbf{U}[t] - \mathbf{U}[T]\|_F^2}{2\mu_t}.
\end{aligned} \tag{2.37}$$

Thanks to the separability of g_t, along with its convexity (cf. Lemma 2), one can establish the inequality

$$g_t(\boldsymbol{\psi}_t, \mathbf{U}[T]) - g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1]) \geq \langle \mathbf{U}[T] - \mathbf{U}[t-1], \nabla_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])\rangle. \tag{2.38}$$

Using (2.38), this yields the following upper bound

$$\begin{aligned}
&T\left[\bar{C}_T - \hat{C}_T\right] \\
&= \sum_{t=1}^{T} \left[g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1]) - g_t(\boldsymbol{\psi}_t, \mathbf{U}[T])\right] \leq \sum_{t=1}^{T} \langle \mathbf{U}[t-1] - \mathbf{U}[T], \nabla_{\mathbf{U}} g_t(\boldsymbol{\psi}_t, \mathbf{U}[t-1])\rangle.
\end{aligned} \tag{2.39}$$

Substituting (2.37) into (2.39), and combining with (2.36), leads to

$$\begin{aligned}
T\left[\bar{C}_T - \hat{C}_T\right] &\leq \frac{\|\mathbf{U}[0] - \mathbf{U}[T]\|_F^2}{2\mu_1} \\
&+ \sum_{t=1}^{T} \left(\frac{1}{2\mu_{t+1}} - \frac{1}{2\mu_t}\right) \|\mathbf{U}[t-1] - \mathbf{U}[T]\|_F^2 + \frac{B^2}{2} \sum_{t=1}^{T} \frac{1}{\mu_t t^2}.
\end{aligned} \tag{2.40}$$

Regarding the first term in the right hand side of (2.40), it can be further bounded by

$$\begin{aligned}
&\frac{\|\mathbf{U}[0]-\mathbf{U}[T]\|_F^2}{2\mu_1}\\
&=\frac{1}{2\mu_1}\|\mathbf{U}[0]-\mathbf{U}[1]+\mathbf{U}[1]-\mathbf{U}[2]+\cdots+\mathbf{U}[T-1]-\mathbf{U}[T]\|_F^2\\
&\leq\frac{1}{2\mu_1}(\|\mathbf{U}[0]-\mathbf{U}[1]\|_F+\cdots+\|\mathbf{U}[T-1]-\mathbf{U}[T]\|_F)^2\\
&=\frac{1}{2\mu_1}\left(\sum_{t=1}^{T}\|\mathbf{U}[t]-\mathbf{U}[t-1]\|_F\right)^2\\
&\leq\frac{1}{2\mu_1}\left(\sum_{t=1}^{T}\frac{B}{t}\right)^2\\
&\leq\frac{B^2}{2\mu_1}(\ln(T)+1)^2 \qquad (2.41)
\end{aligned}$$

where the first inequality follows from the triangle inequality, while the last two inequalities are due to Lemma 3 and the property of harmonic series, respectively. Upon choosing a constant step size $\mu_t=\mu$, the last term in (2.40) can be bounded by [56]

$$\frac{B^2}{2\mu}\sum_{t=1}^{T}\frac{1}{t^2}\leq\frac{5B^2}{6\mu} \qquad (2.42)$$

and after some algebra one arrives at

$$\bar{C}_T-\hat{C}_T\leq\frac{B^2(\ln(T)+1)^2}{2\mu T}+\frac{5B^2}{6\mu T}$$

which completes the proof of Proposition 2.

2.6 Numerical Tests

Performance of the novel online categorical subspace learning schemes is assessed in this section via simulated tests on both synthetic and real-world datasets. The real ones include: (D1) “King-Rook versus King-Pawn” chess-game dataset [57]; and (D2) “Movie-Lens” user-movie rating dataset [58].

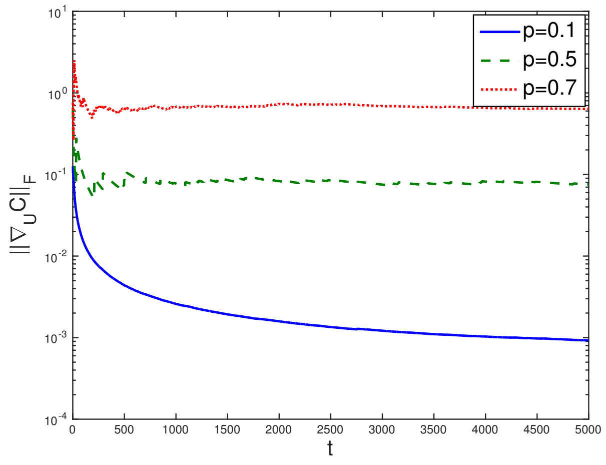


Figure 2.2: Empirical gradient-norm of (P3) versus time for synthetic data under variable $\%$ of misses $(1-p)$.

2.6.1 Synthetic data

Synthetic categorical data $\{\mathbf{y}_t\}_{t=1}^T$ with $D = 25$ across $T = 5,000$ time instants are generated after quantizing the real-valued process $\{\mathbf{x}_t = \mathbf{U}\boldsymbol{\psi}_t\}_{t=1}^T$ to the alphabet $\mathcal{S} := \{1, \ldots, 5\}$. The underlying low-dimensional sketch is drawn equiprobably from two populations, namely $\psi_{i,t} \sim \mathcal{N}(-1, 0.04)$ for the first class; and $\psi_{i,t} \sim \mathcal{N}(+1, 0.04)$ for the second class. Matrix $\mathbf{U} \in \mathbb{R}^{D\times d}$ is generated with entries drawn from the standardized normal distribution. Uniform quantizer is adopted with thresholds $\eta_j := \frac{-J+1+2j}{J-1}x_{\max},\ j = 0, 1, \ldots, J-1$, where $x_{\max}$ denotes the maximum absolute entry of $\mathbf{x}_t$. To simulate the missing entries, a subset of entries are dropped uniformly at random with probability $1-p$.

Throughout the tests a constant step size $\mu_t = 0.01$ is adopted for the subspace update, and the rank controlling parameter is set to $\lambda = 0.1$. The results are averaged over 100 independent trials.

Convergence of Algorithm 1 under various percentages of missing data is demonstrated in Fig. 2.2 depicting the empirical gradient-norm (w.r.t. $\mathbf{U}$) of (P3) over time. It is evident that after about $1,200$ iterations, the online algorithm with random initialization attains a stationary point of (P2). To highlight the merits of the novel scheme, the batch majorization-minimization (MM) scheme of [4] is also implemented. In essence, MM relies on the Logit model with binary data ($J = 2$), and thus one needs first to obtain binary categorical data to make it

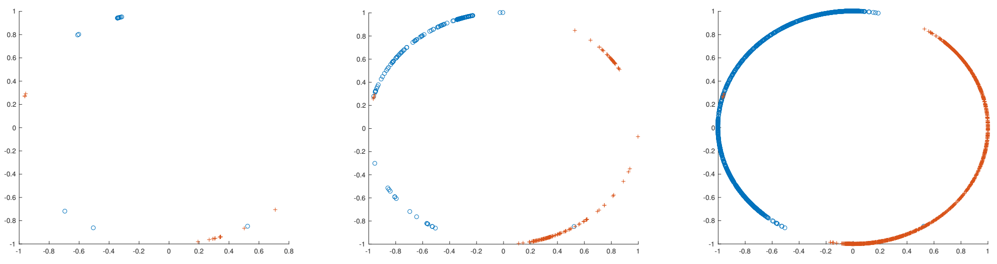


Figure 2.3: Time evolution of the two-dimensional (normalized) sketch obtained using the Probit model at (a) $t = 30$; (b) $t = 300$; and, (c) $t = 3,000$, when $d = 2$, and $D = 15$. The 'x' and 'o' markers are for two different classes. As time goes by, the classes become more separable.

operational. Setting $d = 8$, the low-dimensional sketch returned by both algorithms is used to classify the data using a linear SVM classifier. The resulting runtime as well as the classification error (fraction of miss-classified data) for our scheme and MM are listed in Table 2.1 for a fraction of absent entries. It is apparent that our online scheme exhibits considerable advantage in runtime and accuracy over the batch MM scheme, while also offering real-time sketching and classification of data 'on the fly.'

To further illustrate the operation of real-time sketching, we consider data from the binary quantization model $y_{i,t} = \text{sign}(\mathbf{u}_i^\top \boldsymbol{\psi}_t + v_{i,t})$ with $v_{i,t} \sim \mathcal{N}(0, 0.01)$, and $\mathbf{U} \in \mathbb{R}^{D\times d}$ generated from the standardized normal distribution. The true two-dimensional sketch $\boldsymbol{\psi}_t \in \mathbb{R}^2$ is drawn equiprobably as $[1, 1]^\top$ for the first class, and $[-1, -1]^\top$ for the second class. The normalized sketch $\{\hat{\boldsymbol{\psi}}_\tau\}_{\tau=1}^t$ obtained from the proposed algorithm is depicted in Fig. 2.3 at different time instants $t = 30, 300, 3000$. The number of data points in sequential acquisition (which corresponds to the time instant), coincides with the iteration index. The sketch corresponding two different classes are shown with 'x' and 'o' markers. Apparently, the sketches obtained at the very beginning cannot be well separated as the latent subspace has not been quite learned. However, as more data points are processed, the latent subspace is learned more accurately, and consequently the later data points are assigned to the correct classes.

The Tobit model in Section II.B deals with censored data and is not directly comparable with the Probit and Logit models. To assess the performance of Tobit, synthetic analog-valued data are generated based on the bilinear model $\mathbf{x}_t = \mathbf{U}\boldsymbol{\psi}_t + \mathbf{v}_t$, with $D = 100$, $d = 20$, $T = 200$. Entries of $\mathbf{U}$ and $\mathbf{v}_t$ are drawn from an i.i.d. standardized normal distribution, and

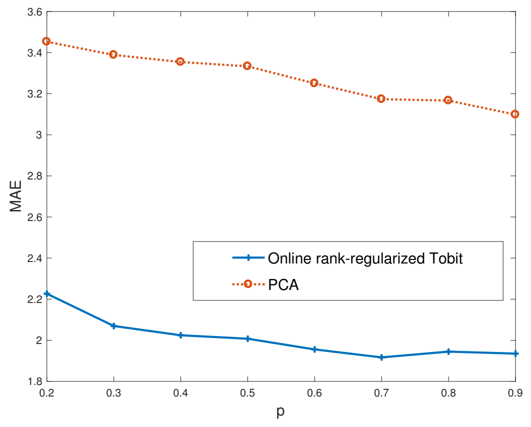
	online CSL	
p	runtime (sec)	classification error (%)
0.1	3.0333	42.17
0.3	2.5925	17.20
0.5	2.7029	4.76
0.7	2.8967	2.18
	MM [4]	
0.1	8.1267	43.86
0.3	6.4973	32.62
0.5	7.5499	17.01
0.7	9.2077	8.31

Table 2.1: Runtime (seconds) and classification error comparison of the proposed online CSL scheme against the batch MM for synthetic data under variable fraction of misses $1-p$.

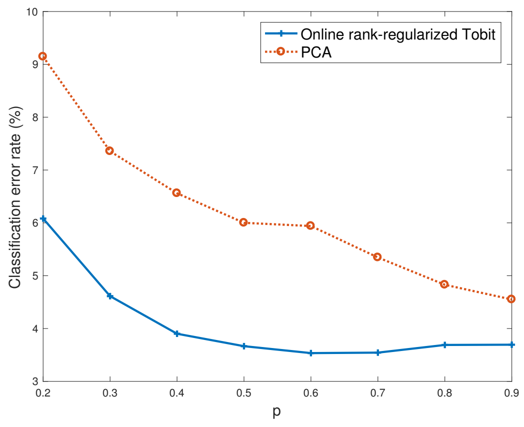
the sketch is also picked as $\boldsymbol{\psi}_t = \mathbf{1}$ for the first class and $\boldsymbol{\psi}_t = -\mathbf{1}$ for the second class. The data is then censored by choosing the thresholds $\eta_l = -1$, $\eta_u = 1$. Both the proposed online Tobit scheme and the conventional PCA are run to return the sketch. Figure 2.4(a) illustrates the mean-absolute-error $\mathrm{MAE} = 1/T \sum_{t=1}^{T} \|\mathbf{y}_t - \hat{\mathbf{y}}_t\|_1$, which clearly shows the performance gain offered by the Tobit scheme in terms of interpolation accuracy. SVM classification [41] is then carried out, and the accuracy is plotted in Fig. 2.4(b) for various fraction of misses. One can observe that the novel CSL scheme improves classification accuracy consistently by about $30\% - 50\%$ relative to the conventional PCA, which in turn indicates the importance of taking censoring into consideration for feature extraction.

2.6.2 Classification of chess games

In this experiment, we considered the chess-game dataset "King-Rook versus King-Pawn" acquired across $T = 3,196$ scenarios, each with $D = 35$ binary ($J = 2$) data signifying nominal attributes. The online sketch returned by Algorithm 1 is used to group games in two classes, namely "white-can-win" and "white-cannot-win," upon averaging the classification outcomes over 100 independent runs. Both Probit and Logit models are tested. As evidenced by Fig. 2.5(a) with 90% random misses ($p = 0.1$), our novel approach achieves considerable runtime advantage over the MM scheme for sketching the partial data, especially when the dimension of the latent subspace is in the order of a few dozens. With the low-dimensional sketch at hand,

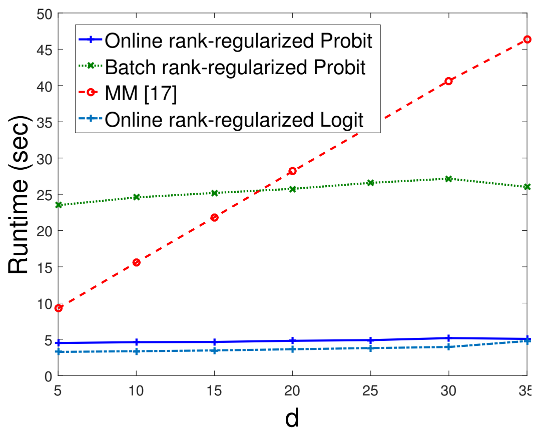


(a)

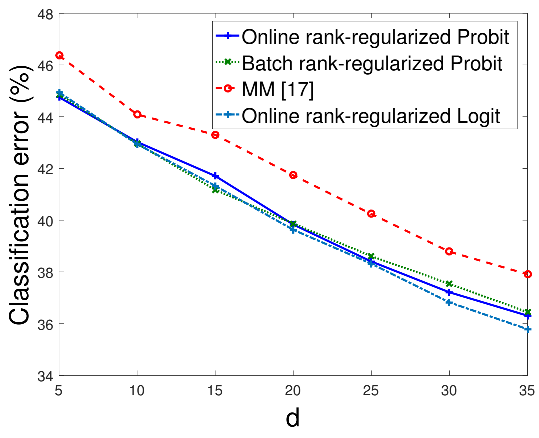


(b)

Figure 2.4: (a) MAE and (b) SVM classification error of CSL scheme with Tobit versus the conventional PCA under variable compression rate p when $d = 20$, $D = 100$, and $T = 200$.



(a)



(b)

Figure 2.5: (a) Runtime and (b) LS classification error of the novel CSL scheme versus the MM scheme for the "King-Rook versus King-Pawn" dataset under variable dimension d when $p = 0.1$, $D = 35$, and $T = 3,196$.

LS classification [41] is performed, and the resultant error is plotted in Figure 2.5(b) under different compression ratios. Our novel CSL-based schemes consistently improve the classification accuracy by about 5% relative to MM, indicating that the adopted models better match

	ML-online				MM [17]	
	epochs =3		epochs =6			
p	runtime	MAE	runtime	MAE	runtime	MAE
0.7	4.5980	0.7832	8.5167	0.7566	40.926	0.7872
0.8	4.5458	0.7745	7.9642	0.7534	40.568	0.7903
0.9	4.7814	0.7724	7.7265	0.7436	35.700	0.7874

Table 2.2: Runtime and MAE comparison of the proposed scheme against the batch MM scheme under various p and different number of epochs for the Movie-Lens dataset with $d = 6$.

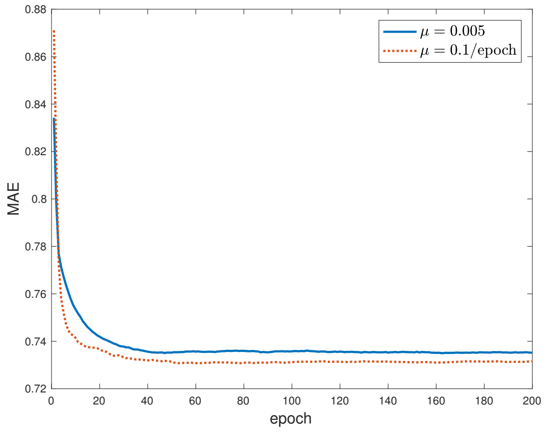


Figure 2.6: MAE versus epochs under different settings of step size.

the considered real-world dataset. In addition, the Probit and Logit models provide similar performance when d is small, while Logit slightly outperforms Probit when d is large. Note that our schemes use only a *single* epoch over the dataset.

2.6.3 Interpolation of "Movie-Lens" dataset

The "Movie-Lens" dataset (D2) is considered to evaluate the interpolation capability of the novel CSL scheme. This dataset originally contains discrete ratings with values in $\mathcal{S} := \{\infty, \infty.\triangledown, \in \ldots, \triangledown\}$ given by $D = 671$ users for $T = 9,066$ movies [58]. To test the proposed method, as well as the MM scheme all ratings were rounded to have only integer values. Note that the time dimension here indexes the released movies over time. To highlight the merits of the novel CSL schemes, a fraction p of the ratings was randomly sampled as training data to learn the latent subspace, and sketching was performed using our scheme and the MM one. Dimension $d = 6$ is selected for the latent subspace. Due to the small size of the training dataset, a single pass would lead to unsatisfactory learning accuracy when initialized randomly. Hence, to improve the ability of our scheme to learn the subspace, multiple epochs were allowed over the data, where the first pass was initialized randomly, and the resulting subspace formed the initial value for the next round, and so on. The resulting subspace and sketch are then used to interpolate the missing ratings. The runtime and MAE are listed in Table 2.2. It can be verified that the novel approach outperforms the MM scheme in terms of both runtime and prediction accuracy. For instance, when 30% of ratings are missing, with six epochs over the data, our scheme offers around 5% gain in prediction accuracy in nearly five times lower runtime.

Finally, we study the sensitivity of the CSL to the number of epochs, and the rank penalty parameter λ. For the Probit model, the MAE is depicted in Fig. 2.6 as epochs increase. Upon choosing a constant step size the MAE decays quickly for the first few epochs, and after almost 40 epochs it converges. Similar behavior is observed for diminishing step size.

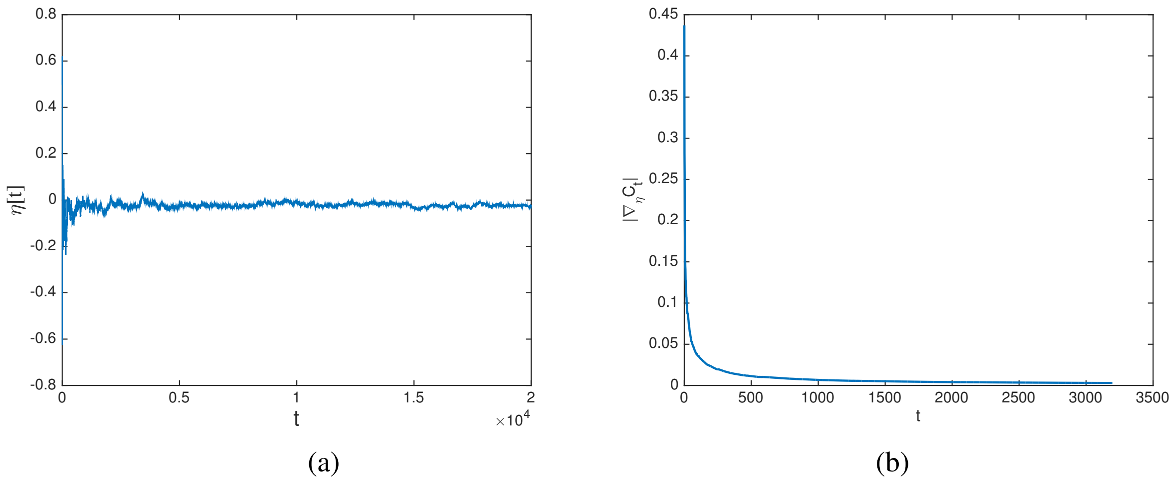


Figure 2.7: Quantization threshold convergence; (left) threshold evolution, and (right) threshold gradient absolute value evolution for chess data.

2.6.4 Threshold adaptation

In this section, convergence and effectiveness of our quantization threshold adaptation is tested for the binary synthetic data described in Sec. 2.6.1. It is observed from Fig. 2.7(a) that by

	online CSL		
p	Runtime (sec)	MAE	classification error (%)
0.6	4.4117	0.3464	6.57
0.7	4.4146	0.3341	6.02
0.8	4.4782	0.2910	4.64
0.9	5.8252	0.2792	4.07
	online CSL with threshold adaptation		
0.6	4.8325	0.2967	6.32
0.7	4.7555	0.2846	5.19
0.8	4.6931	0.2737	4.52
0.9	5.1522	0.2668	3.69

Table 2.3: MAE and classification accuracy comparison of the novel CSL scheme with, and without threshold adaptation, under various p for binary synthetic data when $d = 5$, $D = 20$, and $T = 5,000$.

learning η, the threshold approaches the ground-truth value of $\eta = 0$. The interpolation error as well as the SVM-classification error using the resulting sketch are reported in Table 2.3. Clearly, the threshold adaptation improves the interpolation accuracy by about 17% relative to the CSL scheme that uses the fixed threshold $\eta = 0.5$.

Threshold adaptation is also evaluated on the real chess-game data classification. The performance reported in Table 2.4 shows again 3.7% accuracy improvement relative to the non-adaptive scheme. It is also empirically observed in Fig. 2.7(b) that with the joint quantization threshold and CSL, the threshold iterates converge to a stationary point of the nuclear-norm regularized ML estimator.

2.7 Conclusions and Future Directions

Effective sketching approaches were developed in this chapter for large-scale categorical data that are incomplete and streaming. Low-dimensional Probit, Tobit and Logit models were considered and learned, using a maximum likelihood approach regularized with a surrogate of the nuclear norm. Leveraging separability of this regularizer, and employing stochastic alternating minimization, online algorithms were subsequently developed to sketch the data 'on the fly.' The resultant learning task refines the latent subspace upon arrival of a new datum, and

	online CSL		
p	Runtime (sec)	MAE	classification error (%)
0.6	1.5521	0.7751	24.62
0.7	1.7344	0.7740	24.59
0.8	1.7949	0.7736	24.52
0.9	2.1000	0.7729	24.36
	online CSL with threshold adaptation		
0.6	1.8037	0.7725	23.73
0.7	2.2913	0.7724	23.67
0.8	2.1271	0.7729	23.31
0.9	2.2210	0.7708	23.16

Table 2.4: MAE and classification accuracy comparison of the CSL scheme with, and without threshold adaptation, under various p for the chess-game dataset when $d = 5$, $D = 35$, and $T = 3,196$.

then forms the sketch by projecting the imputed datum onto the latent subspace. This leads to first-order, lightweight, and parallelized iterations. The quantization thresholds are also learned along with the subspace to enhance the modeling flexibility. Performance of the novel algorithms was assessed for both infinite and finite data streams, where for the former asymptotic convergence was established, while for the latter sublinear regret bounds were derived. Simulated tests were carried out on both synthetic and real datasets to confirm the efficacy of the novel schemes for real-time movie recommendation and chess-game classification tasks.

There are still intriguing questions beyond the scope of the present study, that are worth pursuing as future research. One direction pertains to utilizing kernels for nonlinear subspace modeling in an online and computationally efficient fashion. Improving robustness of the categorical subspace learning for dynamic environments with time-varying subspaces is another important avenue to explore. It is also important to extend the proposed CSL scheme to scenarios (the case in recommender systems) where both the ambient dimension (D) and time (T) can possibly increase over time.

2.8 Appendix

2.8.1 Proof of Lemma 1

Assuming $y_{i,t} = \eta_j$ without loss of generality, gradient and Hessian are first derived in closed form

$$\nabla_{\mathbf{u}_i} g_t^{(\text{probit})}\big(\boldsymbol{\psi}_t, \mathbf{U}\big) = -\sigma^{-1}\left[\frac{\phi(z^i_{j,t-1}) - \phi(z^i_{j+1,t-1})}{\mathrm{Q}(z^i_{j,t-1}) - \mathrm{Q}(z^i_{j+1,t-1})}\right]\boldsymbol{\psi}_t + \frac{\lambda}{t}\mathbf{u}_i \tag{2.43}$$

$$\begin{aligned}\nabla^2_{\mathbf{u}_i} g_t^{(\text{probit})}\big(\boldsymbol{\psi}_t, \mathbf{U}\big) = \sigma^{-2}\Bigg\{&\left[\frac{\phi(z^i_{j,t-1}) - \phi(z^i_{j+1,t-1})}{\mathrm{Q}(z^i_{j,t-1}) - \mathrm{Q}(z^i_{j+1,t-1})}\right]^2 \\ &- \frac{z^i_{j,t-1}\phi(z^i_{j,t-1}) - z^i_{j+1,t-1}\phi(z^i_{j+1,t-1})}{\Big[\mathrm{Q}(z^i_{j,t-1}) - \mathrm{Q}(z^i_{j+1,t-1})\Big]}\Bigg\}\boldsymbol{\psi}_t\boldsymbol{\psi}_t^T + \frac{\lambda}{t}\mathbf{I}\end{aligned} \tag{2.44}$$

where $z^i_{j,t-1} := \sigma^{-1}(\eta_j - \mathbf{u}_i^\top[t-1]\boldsymbol{\psi}_t)$. Let us also define

$$r_j := -\frac{\phi(z^i_{j,t-1}) - \phi(z^i_{j+1,t-1})}{\mathrm{Q}(z^i_{j,t-1}) - \mathrm{Q}(z^i_{j+1,t-1})} = \frac{1}{\mathrm{Q}(z^i_{j,t-1}) - \mathrm{Q}(z^i_{j+1,t-1})}\int_{z^i_{j,t-1}}^{z^i_{j+1,t-1}} \epsilon\phi(\epsilon)d\epsilon. \tag{2.45}$$

Since $z^i_{j,t-1} < z^i_{j+1,t-1}$, we have

$$\begin{aligned} z^i_{j,t-1}(\mathrm{Q}\left(z^i_{j,t-1}\right) - \mathrm{Q}(z^i_{j+1,t-1})) &\le \int_{z^i_{j,t-1}}^{z^i_{j+1,t-1}} \epsilon\phi(\epsilon)d\epsilon \\ &\le z^i_{j+1,t-1}(\mathrm{Q}\left(z^i_{j,t-1}\right) - \mathrm{Q}(z^i_{j+1,t-1}))\end{aligned} \tag{2.46}$$

and therefore,

$$r_j \in [z^i_{j,t-1}, z^i_{j+1,t-1}] \le \sigma^{-1}(\eta_j - \eta_{j-1}). \tag{2.47}$$

Hence, one can simply bound the gradient as $\|\nabla_{\mathbf{u}_i} g_t\big(\boldsymbol{\psi}_t, \mathbf{U}\big)\|_2 \le \big\|(\eta_{J-1} - \eta_1)\boldsymbol{\psi}_t/\sigma^2 + \lambda\mathbf{u}_i/t\big\|_2$. Resorting to the triangle inequality, we obtain

$$\|\nabla_{\mathbf{u}_i} g_t^{(\text{probit})}\big(\boldsymbol{\psi}_t, \mathbf{U}\big)\|_2 \le \delta_1\|\boldsymbol{\psi}_t\|_2 + \frac{\lambda}{t}\|\mathbf{u}_i\|_2 \tag{2.48}$$

where $\delta_1 := \Delta/\sigma^2$, and $\Delta := \eta_{J-1} - \eta_0$ is the quantization range.

Likewise, we have

$$\begin{aligned}&-\frac{z^i_{j,t-1}\phi(z^i_{j,t-1})-z^i_{j+1,t-1}\phi(z^i_{j+1,t-1})}{\left[\mathrm{Q}(z^i_{j,t-1})-\mathrm{Q}(z^i_{j+1,t-1})\right]}\\ &=1-\frac{1}{\mathrm{Q}(z^i_{j,t-1})-\mathrm{Q}(z^i_{j+1,t-1})}\int_{z^i_{j,t-1}}^{z^i_{j+1,t-1}}\epsilon^2\phi(\epsilon)d\epsilon\le 1\end{aligned}$$

which implies that the Hessian can simply be bounded by

$$\|\nabla^2_{\mathbf{u}_i}g_t^{(\text{probit})}\big(\boldsymbol{\psi}_t,\mathbf{U}\big)\|\le\frac{r_j^2+1}{\sigma^2}\|\boldsymbol{\psi}_t\|_2^2+\frac{\lambda}{t}\tag{2.49}$$

and thus,

$$\|\nabla^2_{\mathbf{u}_i}h_t^{(\text{probit})}\big(\boldsymbol{\psi}_t,\mathbf{U}\big)\|\le\delta_2\|\boldsymbol{\psi}_t\|_2^2+\frac{\lambda}{t}\tag{2.50}$$

where $\delta_2 := (\Delta^2/\sigma^2+1)/\sigma^2$. Hence, the compactness assumption (as2) implies that the gradient and Hessian are bounded. The differentiability of g_t then leads to Lipschitz continuity of g_t and ∇g_t.

2.8.2 Proof of Lemma 2

According to the gradient expression in (2.17), the Hessian for the Probit cost function can be written as

$$\nabla^2_{\mathbf{u}_i}g_t^{(\text{probit})}\big(\boldsymbol{\psi}_t,\mathbf{U}\big)=\left\{\left[\frac{f(\mathbf{u}_i,\boldsymbol{\psi}_t)}{w(\mathbf{u}_i,\boldsymbol{\psi}_t)}\right]^2-\frac{m(\mathbf{u}_i,\boldsymbol{\psi}_t)}{w(\mathbf{u}_i,\boldsymbol{\psi}_t)}\right\}\boldsymbol{\psi}_t\boldsymbol{\psi}_t^\top+\frac{\lambda}{t}\mathbf{I}\tag{2.51}$$

where

$$\begin{aligned}m(\mathbf{u}_i,\boldsymbol{\psi}_t)&:=z^i_{j,t-1}\phi(z^i_{j,t-1})-z^i_{j+1,t-1}\phi(z^i_{j+1,t-1})\\ f(\mathbf{u}_i,\boldsymbol{\psi}_t)&:=\phi(z^i_{j,t-1})-\phi(z^i_{j+1,t-1})\\ w(\mathbf{u}_i,\boldsymbol{\psi}_t)&:=\mathrm{Q}\left(z^i_{j,t-1}\right)-\mathrm{Q}(z^i_{j+1,t-1})\end{aligned}$$

From (2.47) and the definition of $m(\mathbf{u}_i[t-1], \boldsymbol{\psi}_t)$, we have

$$z^i_{j,t-1} f(\mathbf{u}_i, \boldsymbol{\psi}_t) \leq m(\mathbf{u}_i, \boldsymbol{\psi}_t) \leq z^i_{j+1,t-1} f(\mathbf{u}_i, \boldsymbol{\psi}_t). \tag{2.52}$$

If $r_j > 0$, then $z^i_{j+1,t-1} > 0$, which in combination with (2.52) yields

$$\left[\frac{f(\mathbf{u}_i, \boldsymbol{\psi}_t)}{w(\mathbf{u}_i, \boldsymbol{\psi}_t)}\right]^2 - \frac{m(\mathbf{u}_i, \boldsymbol{\psi}_t)}{w(\mathbf{u}_i, \boldsymbol{\psi}_t)} \geq r_j^2 - z^i_{j,t-1} r_j = r_j(r_j - z^i_{j,t-1}) \geq 0. \tag{2.53}$$

Similarly, if $r_j < 0$, it follows that

$$\left[\frac{f(\mathbf{u}_i, \boldsymbol{\psi}_t)}{w(\mathbf{u}_i, \boldsymbol{\psi}_t)}\right]^2 - \frac{m(\mathbf{u}_i, \boldsymbol{\psi}_t)}{w(\mathbf{u}_i, \boldsymbol{\psi}_t)} \geq r_j^2 - z^i_{j+1,t-1} r_j = r_j(r_j - z^i_{j+1,t-1}) \geq 0. \tag{2.54}$$

Clearly (2.53) and (2.54) imply that the Hessian matrix in (2.51) is positive definite. Hence, the entry-wise cost $g_t(\cdot)$ is convex w.r.t. $\mathbf{u}_i$. Likewise, due to its symmetry w.r.t. $\mathbf{u}_i$ and $\boldsymbol{\psi}_t$, the cost $g_t(\cdot)$ is convex w.r.t. $\boldsymbol{\psi}_t$.

For the binary Logit model, the Hessian of the function can be represented as (cf. (2.17))

$$\begin{aligned}
\nabla^2_{\mathbf{u}_i} g_t^{(\text{logit})}\big(\boldsymbol{\psi}_t, \mathbf{U}\big) &= \left\{\frac{(2y_{i,t}-1)^2 \exp(\mathbf{u}_i^\top \boldsymbol{\psi}_t)}{1+\exp((2y_{i,t}-1)\mathbf{u}_i^\top \boldsymbol{\psi}_t)}\right\} \boldsymbol{\psi}_t \boldsymbol{\psi}_t^\top + \frac{\lambda}{t}\mathbf{I} \\
&= \left\{\frac{\exp(\mathbf{u}_i^\top \boldsymbol{\psi}_t)}{1+\exp((2y_{i,t}-1)\mathbf{u}_i^\top \boldsymbol{\psi}_t)}\right\} \boldsymbol{\psi}_t \boldsymbol{\psi}_t^\top + \frac{\lambda}{t}\mathbf{I}
\end{aligned} \tag{2.55}$$

where the last equation comes from the fact that $|2y_{i,t} - 1| = 1$. It is clear that

$$\frac{\exp(\mathbf{u}_i^\top \boldsymbol{\psi}_t)}{1+\exp((2y_{i,t}-1)\mathbf{u}_i^\top \boldsymbol{\psi}_t)} > 0 \tag{2.56}$$

and hence $\nabla^2_{\mathbf{u}_i} g_t^{(\text{logit})}\big(\boldsymbol{\psi}_t, \mathbf{U}\big) \succ \mathbf{0}$. Likewise, the Hessian matrix of $\boldsymbol{\psi}$ for a fixed subspace $\mathbf{U}$ is also positive definite because the objective function is symmetric with respect to $\mathbf{u}_i$ and $\boldsymbol{\psi}_t$. Hence, the entry-wise cost function is per-block convex in terms of $\mathbf{u}_i$ and $\boldsymbol{\psi}_t$.

For the Tobit-II model in (2.19), the gradient looks similar to that of the Probit model for $y_{i,t} \in (\eta_l, \eta_u)$, and the only difference appears in the threshold values, which will not influence convexity of the function. In fact, for $y_{i,t} = \eta_u$ or $y_{i,t} = \eta_l$, we arrive at

$$\nabla^2_{\mathbf{u}_i} g_t^{(\mathrm{tobit-II})}(\boldsymbol{\psi}_t, \mathbf{U}) = \frac{1}{\sigma^2}\boldsymbol{\psi}_t\boldsymbol{\psi}_t^\top + \frac{\lambda}{t}\mathbf{I} \tag{2.57}$$

which is positive definite. Likewise, the Hessian matrix of $\boldsymbol{\psi}$ for a fixed $\mathbf{U}$ is also positive definite due to the symmetry of $\mathbf{u}_i$ and $\boldsymbol{\psi}_t$. Hence, the entry-wise cost is per-block convex in terms of $\mathbf{u}_i$ and $\boldsymbol{\psi}_t$.

2.8.3 Proof of Lemma 3

First, observe that $\nabla\bar{C}_t(\mathbf{U}[t]) = \nabla\bar{C}_{t+1}(\mathbf{U}[t+1]) = \mathbf{0}$ by construction of the algorithm. Meanwhile, since $\bar{C}_t(\mathbf{U})$ is strongly convex (cf. Lemma 2), the mean-value theorem implies

$$\bar{C}_t(\mathbf{U}[t+1]) \geq \bar{C}_t(\mathbf{U}[t]) + \frac{\rho}{2}\big\|\mathbf{U}[t+1] - \mathbf{U}[t]\big\|_F^2$$
$$\bar{C}_{t+1}(\mathbf{U}[t]) \geq \bar{C}_{t+1}(\mathbf{U}[t+1]) + \frac{\rho}{2}\big\|\mathbf{U}[t+1] - \mathbf{U}[t]\big\|_F^2$$

where ρ denotes the strong convexity constant of $\bar{C}_t(\mathbf{U}[t+1])$. Upon defining the function $\nu_t(\mathbf{U}) := \bar{C}_t(\mathbf{U}) - \bar{C}_{t+1}(\mathbf{U})$, we arrive at

$$\big\|\mathbf{U}[t+1] - \mathbf{U}[t]\big\|_F^2 \leq \frac{1}{\rho}\big|\nu_t(\mathbf{U}[t+1]) - \nu_t(\mathbf{U}[t])\big|. \tag{2.58}$$

Based on the definition of $\bar{C}(\mathbf{U}[t+1])$, we further have

$$\begin{aligned}\nu_t(\mathbf{U}) &= \frac{1}{t}\sum_{\tau=1}^{t} g_\tau(\boldsymbol{\psi}_\tau, \mathbf{U}) - \frac{1}{t+1}\sum_{\tau=1}^{t+1} g_\tau(\boldsymbol{\psi}_\tau, \mathbf{U}) \\ &= \frac{1}{t(t+1)}\sum_{\tau=1}^{t} g_\tau(\boldsymbol{\psi}_\tau, \mathbf{U}) - \frac{1}{t+1} g_{t+1}(\boldsymbol{\psi}_{\tau+1}, \mathbf{U}).\end{aligned} \tag{2.59}$$

Combining Lemma 1 with (2.59), establishes that $\nu_t(\mathbf{U})$ is Lipschitz continuous, and thus

$$\big|\nu_t(\mathbf{U}[t+1]) - \nu_t(\mathbf{U}[t])\big| \leq \frac{\lambda B_u + \delta_1 B_\psi}{t+1}\big\|\mathbf{U}[t+1] - \mathbf{U}[t]\big\|_F \tag{2.60}$$

which after using (2.58) yields

$$\big\|\mathbf{U}[t+1]-\mathbf{U}[t]\big\|_F \leq \frac{\lambda B_u + \delta_1 B_\psi}{(t+1)\rho}. \tag{2.61}$$

Accordingly, Lemma 3 holds with $B := (\lambda B_u + \delta_1 B_\psi)/\rho$.

Chapter 3

Online nonlinear learning in environments with unknown dynamics

3.1 Introduction

Function approximation emerges in various learning tasks such as regression, classification, clustering, dimensionality reduction, as well as reinforcement learning [59–61]. Among them, the emphasis here is placed on supervised functional learning tasks: given a set of data samples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}_{t=1}^{T}$ with $\mathbf{x}_t \in \mathbb{R}^d$ and $y_t \in \mathbb{R}$, the goal is to find a function $f(\cdot)$ such that the discrepancy between each pair of y_t and $f(\mathbf{x}_t)$ is minimized. Typically, such discrepancy is measured by a cost function $\mathcal{C}(f(\mathbf{x}_t), y_t)$, which requires to find $f(\cdot)$ minimizing $\sum_{t=1}^{T} \mathcal{C}(f(\mathbf{x}_t), y_t)$. While this goal is too ambitious to achieve in general, the problem becomes tractable when $f(\cdot)$ is assumed to belong to a reproducing kernel Hilbert space (RKHS) induced by a kernel [59]. Comparable to deep neural networks, functions defined in RKHS can model highly nonlinear relationship, and thus kernel-based methods have well-documented merits for principled function approximation. Despite their popularity, most kernel methods rely on a single pre-selected kernel. Yet, multi-kernel learning (MKL) is more powerful, thanks to its data-driven kernel selection from a given dictionary; see e.g., [60, 62–64], and [65].

In addition to the attractive representation power that can be afforded by kernel methods, several learning tasks are also expected to be performed in an online fashion. Such a need naturally arises when the data arrive sequentially, such as those in online spam detection [66], and time series prediction [67]; or, when the sheer volume of data makes it impossible to carry out data analytics in batch form [68]. This motivates well online kernel-based learning methods that inherit the merits of their batch counterparts, while at the same time allowing efficient online implementation. Taking a step further, the optimal function may itself change over time in environments with *unknown dynamics*. This is the case when the function of interest e.g., represents the state in brain graphs, or, captures the temporal processes propagating over time-varying networks. Especially when variations are due to adversarial interventions, the underlying dynamics are unknown. Online kernel-based learning in such environments remains a largely uncharted territory [68, 69].

In accordance with these needs and desiderata, the *primary goal* of this contribution is an algorithmic pursuit of scalable online MKL in environments with unknown dynamics, along with their associated performance guarantees. Major challenges come from two sources: i) the well-known “curse” of dimensionality in kernel-based learning; and, ii) the defiance of tracking unknown time-varying functions without future information. Regarding i), the representer theorem renders the size of kernel matrices to grow quadratically with the number of data [70], thus the computational complexity to find even a *single* kernel-based predictor is cubic. Furthermore, storage of past data causes memory overflow in large-scale learning tasks such as those emerging in e.g., topology identification of social and brain networks [27, 28], which makes kernel-based methods less scalable relative to their linear counterparts. For ii), most online learning settings presume time invariance or slow dynamics, where an algorithm achieving sub-linear regret incurs on average “no-regret” relative to the *best static* benchmark. Clearly, designing online schemes that are comparable to the *best dynamic* solution is appealing though formidably challenging without knowledge of the dynamics [68, 71].

3.1.1 Related works

To put our work in context, we review prior art from the following two aspects.

- **Batch kernel methods**. Kernel methods are known to suffer from the growing dimensionality in large-scale learning tasks [60]. Major efforts have been devoted to scaling up kernel methods in batch settings. Those include approaches to approximating the kernel matrix using low-rank factorizations [72, 73], whose performance was analyzed in [74]. Recently, random feature (RF) based function estimators have gained popularity since the work of [75] and [76], whose variance has been considerably reduced through an *orthogonality promoting* RF modification [77]. These approaches assume that the kernel is known, a choice crucially dependent on domain knowledge. Enabling kernel selection, several MKL-based approaches have emerged, see e.g., [62–64, 78, 79], and their performance gain has been documented relative to their single kernel counterparts. However, the aforementioned methods are designed for batch settings, and are either intractable or become less efficient in online setups. When the sought functions vary over time and especially when the dynamics are unknown (as in adversarial settings), batch schemes fall short in tracking the optimal function estimators.

- **Online (multi-)kernel learning**. Tailored for streaming large-scale datasets, online kernel-based learning methods have gained due popularity. To deal with the growing complexity of online kernel learning, successful attempts have been made to design *budgeted* kernel learning algorithms, including techniques such as support vector removal [68, 80], and support vector merging [81]. Maintaining an affordable budget, online multi-kernel learning (OMKL) methods have been reported for online classification [69, 82, 83], and regression [84, 85]. Devoid of the need for budget maintenance, online kernel-based learning algorithms based on RF approximation [75] have been developed in [86–88], but only with a single pre-selected kernel. More importantly, existing kernel-based learning approaches implicitly presume a *static* environment, where the benchmark is provided through the best static function (a.k.a. static regret) [53]. However, static regret is not a comprehensive metric for dynamic settings, where the optimal kernel also varies over time and the dynamics are generally unknown as with adversarial settings.

3.1.2 Our contributions

The present chapter develops an adaptive online MKL algorithm, capable of learning a nonlinear function from sequentially arriving data samples. Relative to prior art, our contributions can be summarized as follows.

c1) For the first time, RFs are employed for scalable online MKL tackled by a weighted combination of advices from an ensemble of experts - an innovative cross-fertilization of online learning to MKL. Performance of the resultant algorithm (abbreviated as **Raker**) is benchmarked by the best time-invariant function approximant via static regret analysis.

c2) A novel adaptive approach (termed **AdaRaker**) is introduced for scalable online MKL in environments with unknown dynamics. AdaRaker is a hierarchical ensemble learner with scalable RF-based modules that provably yields sub-linear dynamic regret, so long as the accumulated variation grows sub-linearly with time.

c3) The novel algorithms are compared with competing alternatives for online nonlinear regression on both synthetic and real datasets. The tests corroborate that Raker and AdaRaker exhibit attractive performance in both accuracy and scalability.

Outline. Section 3.2 presents preliminaries, and states the problem. Section 3.3 develops the Raker for online MKL in static environments, and Section 3.4 develops its adaptive version for online MKL in environments with unknown dynamics. Section 5.6 reports numerical tests with both synthetic and real datasets, while conclusions are drawn in Section 3.6.

3.2 Preliminaries and Problem Statement

This section reviews briefly basics of kernel-based learning, to introduce notation and the needed background for our novel online MKL schemes.

Given samples $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_T, y_T)\}_{t=1}^T$ with $\mathbf{x}_t \in \mathbb{R}^d$ and $y_t \in \mathbb{R}$, the function approximation task is to find a function $f(\cdot)$ such that $y_t = f(\mathbf{x}_t) + e_t$, where e_t denotes an error term representing noise or un-modeled dynamics. It is supposed that $f(\cdot)$ belongs to a reproducing kernel Hilbert space (RKHS), namely $\mathcal{H} := \{f | f(\mathbf{x}) = \sum_{t=1}^{\infty} \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t)\}$, where $\kappa(\mathbf{x}, \mathbf{x}_t) : \mathbb{R}^d \times \mathbb{R}^d \rightarrow \mathbb{R}$ is a symmetric positive semidefinite basis (so-termed kernel) function, which measures the similarity between $\mathbf{x}$ and $\mathbf{x}_t$. Among the choices of κ specifying different bases, a popular one is the Gaussian given by $\kappa(\mathbf{x}, \mathbf{x}_t) := \exp[-\|\mathbf{x} - \mathbf{x}_t\|^2/(2\sigma^2)]$. A kernel is reproducing if it satisfies $\langle \kappa(\mathbf{x}, \mathbf{x}_t), \kappa(\mathbf{x}, \mathbf{x}_{t'}) \rangle_{\mathcal{H}} = \kappa(\mathbf{x}_t, \mathbf{x}_{t'})$, which in turn induces the RKHS norm $\|f\|_{\mathcal{H}}^2 := \sum_t \sum_{t'} \alpha_t \alpha_{t'} \kappa(\mathbf{x}_t, \mathbf{x}_{t'})$. Consider the optimization problem

$$\min_{f \in \mathcal{H}} \frac{1}{T} \sum_{t=1}^{T} \mathcal{C}(f(\mathbf{x}_t), y_t) + \lambda \Omega \left(\|f\|_{\mathcal{H}}^2 \right) \tag{3.1}$$

where depending on the application, the cost function $\mathcal{C}(\cdot, \cdot)$ can be selected to be, e.g., the least-squares (LS), the logistic or the hinge loss; $\Omega(\cdot)$ is an increasing function; and, $\lambda > 0$ is a regularization parameter that controls overfitting. According to the representer theorem, the optimal solution of (3.1) admits the finite-dimensional form, given by [70]

$$\hat{f}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \kappa(\mathbf{x}, \mathbf{x}_t) := \boldsymbol{\alpha}^\top \mathbf{k}(\mathbf{x}) \tag{3.2}$$

where $\boldsymbol{\alpha} := [\alpha_1, \ldots, \alpha_T]^\top \in \mathbb{R}^T$ collects the combination coefficients, and the $T \times 1$ kernel vector is $\mathbf{k}(\mathbf{x}) := [\kappa(\mathbf{x}, \mathbf{x}_1), \ldots, \kappa(\mathbf{x}, \mathbf{x}_T)]^\top$. Substituting (3.2) into the RKHS norm, we find $\|f\|_{\mathcal{H}}^2 := \sum_t \sum_{t'} \alpha_t \alpha_{t'} \kappa(\mathbf{x}_t, \mathbf{x}_{t'}) = \boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}$, where the $T \times T$ kernel matrix $\mathbf{K}$ has entries $[\mathbf{K}]_{t,t'} := \kappa(\mathbf{x}_t, \mathbf{x}_{t'})$; thus, the functional problem (3.1) boils down to a T-dimensional optimization over $\boldsymbol{\alpha}$, namely

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^T} \frac{1}{T} \sum_{t=1}^{T} \mathcal{C}(\boldsymbol{\alpha}^\top \mathbf{k}(\mathbf{x}_t), y_t) + \lambda \Omega \left(\boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} \right) \tag{3.3}$$

where $\mathbf{k}^\top(\mathbf{x}_t)$ is the tth row of the matrix $\mathbf{K}$. While a scalar y_t is used here for brevity, coverage extends readily to vectors $\{\mathbf{y}_t\}$.

Note that (3.1) relies on: i) a known pre-selected kernel κ; and ii) having $\{\mathbf{x}_t, y_t\}_{t=1}^T$ available in batch form. A key observation here is that the dimension of the variable $\boldsymbol{\alpha}$ in (3.3) grows with time T (or, the number of samples in the batch form), making it less scalable in online implementation. In the ensuing section, an online MKL method will be proposed to select κ as a superposition of multiple kernels, when the data become available online.

3.3 Online MKL in static environments

In this section, we develop an online learning approach that builds on the notion of **ra**ndom features [75, 77], and leverages in a unique way multi-**ker**nel approximation – two tools justifying our acronym **Raker** used henceforth.

3.3.1 RF-based single kernel learning

To cope with the curse of dimensionality in optimizing (3.3), we will reformulate the functional optimization problem (3.1) as a parametric one with the dimension of optimization variables not growing with time. In this way, powerful toolboxes from convex optimization and online learning in vector spaces can be leveraged. We achieve this goal by judiciously using RFs. Although generalizations will follow, this subsection is devoted to RF-based single kernel learning, where basics of kernels, RFs, and online learning will be revisited.

As in [75], we will approximate κ in (3.2) using shift-invariant kernels that satisfy $\kappa(\mathbf{x}_t, \mathbf{x}_{t'}) = \kappa(\mathbf{x}_t - \mathbf{x}_{t'})$. For $\kappa(\mathbf{x}_t - \mathbf{x}_{t'})$ absolutely integrable, its Fourier transform $\pi_\kappa(\mathbf{v})$ exists and represents the power spectral density, which upon normalizing to ensure $\kappa(\mathbf{0}) = 1$, can also be viewed as a probability density function (pdf); hence,

$$\kappa(\mathbf{x}_t - \mathbf{x}_{t'}) = \int \pi_\kappa(\mathbf{v}) e^{j\mathbf{v}^\top(\mathbf{x}_t - \mathbf{x}_{t'})} d\mathbf{v} := \mathbb{E}_\mathbf{v}\left[e^{j\mathbf{v}^\top(\mathbf{x}_t - \mathbf{x}_{t'})}\right] \tag{3.4}$$

where the last equality is just the definition of the expected value. Drawing a sufficient number of D independent and identically distributed (i.i.d.) samples $\{\mathbf{v}_i\}_{i=1}^D$ from $\pi_\kappa(\mathbf{v})$, the ensemble mean in (3.4) can be approximated by the sample average

$$\hat{\kappa}_c(\mathbf{x}_t, \mathbf{x}_{t'}) := \frac{1}{D}\sum_{i=1}^{D} e^{j\mathbf{v}_i^\top(\mathbf{x}_t - \mathbf{x}_{t'})} := \boldsymbol{\zeta}_\mathbf{V}^\dagger(\mathbf{x}_t)\boldsymbol{\zeta}_\mathbf{V}(\mathbf{x}_{t'}) \tag{3.5}$$

where $\mathbf{V} := [\mathbf{v}_1, \ldots, \mathbf{v}_D]^\top \in \mathbb{R}^{D\times d}$, symbol $\dagger$ represents the Hermitian (conjugate-transpose) operator, and $\boldsymbol{\zeta}_\mathbf{V}(\mathbf{x})$ the complex RF vector

$$\boldsymbol{\zeta}_\mathbf{V}(\mathbf{x}) := \frac{1}{\sqrt{D}}\left[e^{j\mathbf{v}_1^\top\mathbf{x}}, \ldots, e^{j\mathbf{v}_D^\top\mathbf{x}}\right]^\top . \tag{3.6}$$

Taking expected values on both sides of (3.5) and using (3.4) yields $\mathbb{E}_\mathbf{v}[\hat{\kappa}_c(\mathbf{x}_t, \mathbf{x}_{t'})] = \kappa(\mathbf{x}_t, \mathbf{x}_{t'})$, which means $\hat{\kappa}_c$ is unbiased. Likewise, $\hat{\kappa}_c$ can be shown consistent since $\mathbb{V}\mathrm{ar}[\hat{\kappa}_c(\mathbf{x}_t, \mathbf{x}_{t'})] \propto D^{-1}$ vanishes as $D \to \infty$. Finding $\pi_\kappa(\mathbf{v})$ requires d-dimensional Fourier transform of κ, generally through numerical integration. For a number of popular kernels however, $\pi_\kappa(\mathbf{v})$ is available in closed form. Taking the Gaussian kernel as an example, where $\kappa_G(\mathbf{x}_t, \mathbf{x}_{t'}) = \exp\left(\|\mathbf{x}_t - \mathbf{x}_{t'}\|_2^2/(2\sigma^2)\right)$, has Fourier transform corresponding to the pdf $\pi_G(\mathbf{v}) = \mathcal{N}(0, \sigma^{-2}\mathbf{I})$.

Instead of the *complex* RFs $\{\boldsymbol{\zeta}_\mathbf{V}(\mathbf{x}_t)\}$ in (3.6) forming the linear kernel estimator $\hat{\kappa}_c$ in (3.5), one can consider its real part $\hat{\kappa}(\mathbf{x}_t, \mathbf{x}_{t'}) := \Re\{\hat{\kappa}_c(\mathbf{x}_t, \mathbf{x}_{t'})\}$ that is also an unbiased estimator of κ. Defining the real RF vector $\mathbf{z}_\mathbf{V}(\mathbf{x}) := [\Re^\top\{\boldsymbol{\zeta}_\mathbf{V}(\mathbf{x}_t)\}, \Im^\top\{\boldsymbol{\zeta}_\mathbf{V}(\mathbf{x}_t)\}]^\top$, this real kernel estimator becomes (cf. (3.5))

$$\hat{\kappa}(\mathbf{x}_t, \mathbf{x}_{t'}) = \mathbf{z}_\mathbf{V}^\top(\mathbf{x}_t)\mathbf{z}_\mathbf{V}(\mathbf{x}_{t'}) \tag{3.7}$$

where the $2D \times 1$ *real* RF vector can be written as

$$\mathbf{z}_{\mathbf{V}}(\mathbf{x}) = \frac{1}{\sqrt{D}} \left[\sin(\mathbf{v}_1^\top \mathbf{x}), \ldots, \sin(\mathbf{v}_D^\top \mathbf{x}), \cos(\mathbf{v}_1^\top \mathbf{x}), \ldots, \cos(\mathbf{v}_D^\top \mathbf{x})\right]^\top . \tag{3.8}$$

Hence, the nonlinear function that is optimal in the sense of (3.1) can be approximated by a linear one in the new $2D$-dimensional RF space, namely (cf. (3.2) and (3.7))

$$\hat{f}^{\mathrm{RF}}(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t \mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_t) \mathbf{z}_{\mathbf{V}}(\mathbf{x}) := \boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}) \tag{3.9}$$

where $\boldsymbol{\theta}^\top := \sum_{\tau=1}^{T} \alpha_\tau \mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_\tau)$ is the new weight vector of size $2D$ whose dimension does not increase with number of data samples T.

While the solution $\hat{f}$ in (3.2) is the superposition of nonlinear functions κ, its RF approximant $\hat{f}^{\mathrm{RF}}$ in (3.9) is a linear function of $\mathbf{z}_{\mathbf{V}}(\mathbf{x})$. As a result, the loss becomes

$$\mathcal{L}_t\big(f(\mathbf{x}_t)\big) := \mathcal{C}(f(\mathbf{x}_t), y_t) + \lambda\Omega\left(\|f\|_{\mathcal{H}}^2\right) = \mathcal{C}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t\big) + \lambda\Omega\left(\|\boldsymbol{\theta}\|^2\right) \tag{3.10}$$

where $\|\boldsymbol{\theta}\|^2 := \sum_t \sum_{t'} \alpha_t \alpha_{t'} \mathbf{z}_{\mathbf{V}}^\top(\mathbf{x}_t) \mathbf{z}_{\mathbf{V}}(\mathbf{x}_{t'}) := \|f\|_{\mathcal{H}}^2$; and the online learning task is

$$\min_{\boldsymbol{\theta} \in \mathbb{R}^{2D}} \sum_{t=1}^{T} \mathcal{L}\left(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t\right), \text{ with } \mathcal{L}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t\big) := \mathcal{C}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t\big) + \lambda\Omega\big(\|\boldsymbol{\theta}\|^2\big). \tag{3.11}$$

Compared with the functional optimization in (3.1), the reformulated problem (3.11) is parametric, and more importantly it involves only optimization variables of fixed size $2D$. We can thus solve (3.11) using the online gradient descent iteration, e.g., [89]. Acquiring $\mathbf{x}_t$ per slot t, its RF $\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t)$ is formed as in (3.8), and $\boldsymbol{\theta}_{t+1}$ is updated 'on the fly,' as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla \mathcal{L}(\boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t) \tag{3.12}$$

where $\{\eta_t\}$ is the sequence of stepsizes that can tune learning rates, and $\nabla \mathcal{L}(\boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t)$ the gradient at $\boldsymbol{\theta} = \boldsymbol{\theta}_t$. Iteration (3.12) provides *a functional update* since $\hat{f}_t^{\mathrm{RF}}(\mathbf{x}) = \boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x})$, but the upshot of involving RFs is that this approximant is in the span of $\{\mathbf{z}_{\mathbf{V}}(\mathbf{x}), \forall \mathbf{x} \in \mathcal{X}\}$. Since $\mathbb{E}[\hat{\kappa}] = \kappa$, we find readily that $\mathbb{E}[\hat{f}^{\mathrm{RF}}] = \hat{f}$; in words, unbiasedness of the kernel approximation ensures that the RF-based function approximant is also unbiased.

Variance-reduced RF. Besides unbiasedness, performance of the RF approximation is also influenced by the variance of RFs. Note that the variance of $\hat{\kappa}$ in (3.7) is of order $\mathcal{O}(D^{-1})$, but its scale can be reduced if $\mathbf{V}$ is formed to have orthogonal rows [77]. Specifically for a Gaussian kernel with bandwidth σ^2, recall that $\mathbf{V} = \sigma^{-1}\mathbf{G}$ in (3.8), where each entry of $\mathbf{G}$ is drawn from $\mathcal{N}(0,1)$. For the variance-reduced orthogonal (O)RF with $D = d$, one starts with Q-R factorization of $\mathbf{V} = \mathbf{QR}$, and uses the $d \times d$ factor $\mathbf{Q}$ along with a diagonal matrix $\mathbf{\Lambda}$, to form [77]

$$\mathbf{V}_{\text{ORF}} = \sigma^{-1}\,\mathbf{\Lambda}\mathbf{Q} \tag{3.13}$$

where the diagonal entries of $\mathbf{\Lambda}$ are drawn i.i.d. from the χ distribution with d degrees of freedom, to ensure unbiasedness of the kernel approximant. For $D > d$, one selects $D = \nu d$ with $\nu > 1$ integer, and generates independently ν matrices each of size $d \times d$ as in (3.13). The final $\mathbf{V}_{\text{ORF}}$ is formed by concatenating these $d \times d$ sub-matrices. The upshot of ORF is that [77] $\mathbb{V}\text{ar}(\hat{\kappa}_{\text{ORF}}(\mathbf{x}_t, \mathbf{x}_{t'})) \leq \mathbb{V}\text{ar}(\hat{\kappa}(\mathbf{x}_t, \mathbf{x}_{t'}))$. As we have also confirmed via simulated tests, ORF-based function approximation can attain a prescribed accuracy with considerably less ORFs than what required by its RF-based counterpart.

The RF-based online single kernel learning scheme in this section presumes that κ is known a priori. Since this is not generally possible, it is prudent to adaptively select kernels by superimposing multiple kernel functions from a prescribed dictionary. This superposition will play a key role in the RF-based online MKL approach presented next.

3.3.2 Raker for online MKL

Specifying the kernel that "shapes" $\mathcal{H}$ is a critical choice for single kernel learning, since different kernels yield function estimates of variable accuracy. To deal with this, combinations of kernels from a prescribed and sufficiently rich dictionary $\{\kappa_p\}_{p=1}^P$ can be employed in (3.1). Each combination belongs to the convex hull $\bar{\mathcal{K}} := \{\bar{\kappa} = \sum_{p=1}^P \bar{\alpha}_p \kappa_p,\ \bar{\alpha}_p \geq 0,\ \sum_{p=1}^P \bar{\alpha}_p = 1\}$, and is itself a kernel [59]. With $\bar{\mathcal{H}}$ denoting the RKHS induced by $\bar{\kappa} \in \bar{\mathcal{K}}$, one then solves (3.1) with $\mathcal{H}$ replaced by $\bar{\mathcal{H}} := \mathcal{H}_1 \bigoplus \cdots \bigoplus \mathcal{H}_P$, where $\{\mathcal{H}_p\}_{p=1}^P$ represent the RKHSs corresponding to $\{\kappa_p\}_{p=1}^P$ [90].

The candidate function $\bar{f} \in \bar{\mathcal{H}}$ is expressible in a separable form as $\bar{f}(\mathbf{x}) := \sum_{p=1}^P \bar{f}_p(\mathbf{x})$, where $\bar{f}_p(\mathbf{x})$ belongs to $\mathcal{H}_p$, for $p \in \mathcal{P} := \{1, \ldots, P\}$. To add flexibility per kernel in our ensuing online MKL scheme, we let wlog $\{\bar{f}_p = w_p f_p\}_{p=1}^P$, and seek functions of the form

$$f(\mathbf{x}) := \sum_{p=1}^{P} \bar{w}_p f_p(\mathbf{x}) \in \bar{\mathcal{H}} \tag{3.14}$$

where $f := \bar{f}/\sum_{p=1}^P w_p$, and the normalized weights $\{\bar{w}_p := w_p/\sum_{p=1}^P w_p\}_{p=1}^P$ satisfy $\bar{w}_p \geq 0$, and $\sum_{p=1}^P \bar{w}_p = 1$. Plugging (3.14) into (3.1), MKL solves the nonconvex problem

$$\min_{\{\bar{w}_p\},\{f_p\}} \frac{1}{T}\sum_{t=1}^{T} \mathcal{C}\left(\sum_{p=1}^{P} \bar{w}_p f_p(\mathbf{x}_t), y_t\right) + \lambda\Omega\left(\left\|\sum_{p=1}^{P} \bar{w}_p f_p\right\|_{\bar{\mathcal{H}}}^2\right) \tag{3.15a}$$

$$\text{s. to } \sum_{p=1}^{P} \bar{w}_p = 1,\ \bar{w}_p \geq 0,\ p \in \mathcal{P} \tag{3.15b}$$

$$f_p \in \mathcal{H}_p,\ p \in \mathcal{P}. \tag{3.15c}$$

If Ω is convex over f, then (3.15a) is biconvex, meaning it is convex wrt $\{f_p\}$ ($\{\bar{w}_p\}$) when $\{\bar{w}_p\}$ ($\{f_p\}$) is given. Leveraging biconvexity, existing batch MKL schemes solve (3.15) via alternating minimization that is known not to scale well with P and T [63,64,90].

To deal with scalability, our novel approach will leverage for the first time (O)RFs in a uniquely principled MKL formulation to end up with an efficient online learning approach. To this end, we will minimize a cost that upper bounds that in (3.15a), namely

$$\min_{\{\bar{w}_p\},\{f_p\}} \frac{1}{T}\sum_{t=1}^{T}\sum_{p=1}^{P} \bar{w}_p\, \mathcal{C}\left(f_p(\mathbf{x}_t), y_t\right) + \lambda \sum_{p=1}^{P} \bar{w}_p\, \Omega\left(\|f_p\|_{\mathcal{H}_p}^2\right) \quad \text{s. to (3.15b) and (3.15c)} \tag{3.16}$$

where Jensen's inequality confirms that under (3.15b) the cost in (3.16) upper bounds that of (3.15a). A key advantage of (3.16) is that its objective is separable across kernel 'atoms.'

We will exploit this separability jointly with the RF-based function approximation per kernel, to formulate our scalable online MKL task as

$$\min_{\{\bar{w}_p\},\{\hat{f}_p^{\mathrm{RF}}\}} \sum_{t=1}^{T}\sum_{p=1}^{P} \bar{w}_p\, \mathcal{L}_t\left(\hat{f}_p^{\mathrm{RF}}(\mathbf{x}_t)\right) \ \text{ s. to (3.15b) and } \hat{f}_p^{\mathrm{RF}} \in \left\{\hat{f}_p(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}_p}(\mathbf{x})\right\} \tag{3.17}$$

where we interchangeably use $\mathcal{L}_t(\hat{f}(\mathbf{x}_t))$ as defined in (3.10) and $\mathcal{L}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t\big)$ as in (3.11). We will efficiently solve (3.17) 'on-the-fly' using our Raker algorithm, and what more, we will provide analytical performance guarantees. Our iterative solution will update separately each $\hat{f}_p^{\mathrm{RF}}$ as in Section 3.3.1 using the scalable (O)RF-based function approximation scheme. Given $\mathbf{x}_t$, an RF vector $\mathbf{z}_p(\mathbf{x}_t)$ will be generated per p from pdf $\pi_{\kappa_p}(\mathbf{v})$ (cf. (3.8)), where we let $\mathbf{z}_p(\mathbf{x}_t) := \mathbf{z}_{\mathbf{V}_p}(\mathbf{x}_t)$ for notational brevity. Hence, for each p and slot t, we have

$$\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{x}_t) = \boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t) \tag{3.18}$$

and as in (3.12), $\boldsymbol{\theta}_{p,t}$ is updated via

$$\boldsymbol{\theta}_{p,t+1} = \boldsymbol{\theta}_{p,t} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t). \tag{3.19}$$

As far as solving for $\bar{w}_{p,t}$, since it resides on a probability simplex (3.15b), our idea is to employ a multiplicative update (a.k.a. exponentiated gradient descent), e.g., [89]. Specifically, the un-normalized weights are found first as

$$w_{p,t+1} = \arg\min_{w_p} \eta\, \mathcal{L}_t\left(\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{x}_t)\right)(w_p - w_{p,t}) + \mathcal{D}_{\mathrm{KL}}(w_p \| w_{p,t}) \tag{3.20}$$

where $\mathcal{D}_{\mathrm{KL}}(w_p \| w_{p,t}) := w_p \log(w_p / w_{p,t})$ is the KL-divergence. It can be readily verified that (3.20) admits the following closed-form update

$$w_{p,t+1} = w_{p,t} \exp\left(-\eta \mathcal{L}_t\left(\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{x}_t)\right)\right) \tag{3.21}$$

where $\eta \in (0,1)$ is a chosen constant that controls the adaptation rate of $\{w_{p,t}\}$. Having found $\{w_{p,t}\}$ as in (3.21), the normalized weights in (3.14) are obtained as $\bar{w}_{p,t} := w_{p,t} / \sum_{p=1}^{P} w_{p,t}$. Update (3.21) is intuitively pleasing because when $\hat{f}_{p,t}^{\mathrm{RF}}$ contributes a larger loss relative to other $\hat{f}_{p',t}^{\mathrm{RF}}$ with $p' \neq p$ at slot t, the corresponding $w_{p,t+1}$ decreases more than the other weights in the next time slot. In other words, a more accurate RF-based approximant tends to play more important role in predicting the upcoming data.

Remark 1. The update (3.21) resembles the online learning paradigm, a.k.a. *online prediction with* (weighted) *expert advices* [91, 92]. Building on but going beyond OMKL in [84], the idea here is to view MKL with *RF-based function approximants* as a weighted combination of

Algorithm 2b Raker for online MKL in static environments

```
1: Input: Kernels $\kappa_p,\ p = 1, \ldots, P$, step size $\eta > 0$, and number of random features $D$.
2: Initialization: $\boldsymbol{\theta}_1 = \mathbf{0}$.
3: for $t = 1, 2, \ldots, T$ do
4:     Receive a streaming datum $\mathbf{x}_t$.
5:     Construct $\mathbf{z}_p(\mathbf{x}_t)$ via (3.8) using $\kappa_p$ for $p = 1, \ldots, P$.
6:     Predict $\hat{f}_t^{\text{RF}}(\mathbf{x}_t) := \sum_{p=1}^{P} \bar{w}_{p,t} \hat{f}_{p,t}^{\text{RF}}(\mathbf{x}_t)$ with $\hat{f}_{p,t}^{\text{RF}}(\mathbf{x}_t)$ in (3.18).
7:     Observe loss function $\mathcal{L}_t$, incur $\mathcal{L}_t(\hat{f}_t^{\text{RF}}(\mathbf{x}_t))$.
8:     for $p = 1, \ldots, P$ do
9:         Obtain loss $\mathcal{L}(\boldsymbol{\theta}_{p,t}^{\top} \mathbf{z}_p(\mathbf{x}_t), y_t)$ or $\mathcal{L}_t(\hat{f}_{p,t}^{\text{RF}}(\mathbf{x}_t))$.
10:        Update $\boldsymbol{\theta}_{p,t+1}$ via (3.19).
11:        Update $w_{p,t+1}$ via (3.21).
12:    end for
13: end for
```

advices from an ensemble of P function approximants (experts). Besides permeating benefits from online learning to MKL, what is distinct here relative to [91, 92] is that each function approximant also performs online learning for self improvement (cf. (3.19)).

In summary, our Raker for static (or slow-varying) dynamics is listed as Algorithm 2b.

Memory requirement and computational complexity. At the t-th iteration, our Raker in Algorithm 2b needs to store a real $2D$ RF vector, and its corresponding weight vector per κ_p. Hence, the memory required is of order $\mathcal{O}(dDP)$. Regarding computational overhead, the per-iteration complexity (e.g., calculating inner products) is again of order $\mathcal{O}(dDP)$. Compared with the complexity of $\mathcal{O}(tdP)$ for OMKL by [84], or, $\mathcal{O}(t^3P)$ when matrix inversion required for the batch MKL, e.g., [65], the Raker is clearly more scalable, as t grows. Even when OMKL is confined to a budget of B past samples, the corresponding complexity of $\mathcal{O}(dBP)$ is comparable to that of Raker. This speaks for Raker's merits, whose performance guarantees will be proved analytically, and also demonstrated by numerical tests to outperform budgeted schemes.

Application examples: Online MKL regression and classification. To appreciate the usefulness of RF-based online MKL, consider first nonlinear regression, where given samples $\{\mathbf{x}_t \in \mathbb{R}^d, y_t \in \mathbb{R}\}_{t=1}^T$, the goal is to find a nonlinear function $f \in \mathcal{H}$, such that $y_t = f(\mathbf{x}_t) + e_t$. The criterion is to minimize the regularized prediction error of y_t, typically using the LS loss

$\mathcal{L}(f(\mathbf{x}_t), y_t) := [y_t - f(\mathbf{x}_t)]^2 + \lambda\|f\|_{\mathcal{H}}^2$, whose gradient is (cf. (3.19))

$$\nabla\mathcal{L}\left(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t\right) = 2(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t) - y_t)\mathbf{z}_p(\mathbf{x}_t) + 2\lambda\boldsymbol{\theta}_{p,t}. \tag{3.22}$$

It is clear that the per iteration complexity of Raker is only related to the dimension of $\mathbf{z}_p(\mathbf{x}_t)$, and does not increase over time.

For nonlinear classification, consider kernel-based perceptron and kernel-based logistic regression, which aim at learning a nonlinear classifier that best approximates either y_t or the pdf of y_t conditioned on $\mathbf{x}_t$. With binary labels $\{\pm 1\}$, the perceptron solves (3.1) with $\mathcal{L}(f(\mathbf{x}_t), y_t) = \max(0, 1 - y_t f(\mathbf{x}_t)) + \lambda\|f\|_{\mathcal{H}}^2$, which equals zero if $y_t = f(\mathbf{x}_t)$, otherwise it equals 1. Raker's gradient in this case is (cf. (3.19))

$$\nabla\mathcal{L}\left(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t\right) = -2y_t\mathcal{C}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t)\mathbf{z}_p(\mathbf{x}_t) + 2\lambda\boldsymbol{\theta}_{p,t}. \tag{3.23}$$

Accordingly, given $\mathbf{x}_t$, logistic regression postulates that $\Pr(y_t = 1|\mathbf{x}_t) = 1/(1+\exp(f(\mathbf{x}_t)))$. Here the gradient of Raker takes the form (cf. (3.19))

$$\nabla\mathcal{L}\left(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t\right) = \frac{2y_t \exp(y_t\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t))}{1 + \exp(y_t\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t))}\mathbf{z}_p(\mathbf{x}_t) + 2\lambda\boldsymbol{\theta}_{p,t}. \tag{3.24}$$

To compare alternatives on equal footing, the numerical tests in Section 5.6 will deal with kernel-based regression and classification.

3.3.3 Static regret analysis of Raker

To analyze the performance of Raker, we assume that the following conditions are satisfied.

(as1) *Per slot t, the loss function* $\mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t)$ *in* (3.11) *is convex w.r.t.* $\boldsymbol{\theta}$.

(as2) *For* $\boldsymbol{\theta}$ *belonging to a bounded set* $\boldsymbol{\Theta}$ *with* $\|\boldsymbol{\theta}\| \leq C_\theta$*, the loss is bounded; that is,* $\mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t) \in [-1, 1]$*, and has bounded gradient, meaning,* $\|\nabla\mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t), y_t)\| \leq L$.

(as3) *Kernels* $\{\kappa_p\}_{p=1}^P$ *are shift-invariant, standardized, and bounded, that is,* $\kappa_p(\mathbf{x}_i, \mathbf{x}_j) \leq 1, \forall \mathbf{x}_i, \mathbf{x}_j$*; and w.l.o.g. they also have bounded entries, meaning* $\|\mathbf{x}\| \leq 1$.

Convexity of the loss under (as1) is satisfied by the popular loss functions including the square loss and the hinge loss. As far as (as2), it ensures that the losses, and their gradients are bounded, meaning they are L-Lipschitz continuous. While boundedness of the losses commonly holds since $\|\boldsymbol{\theta}\|$ is bounded, Lipschitz continuity is also not restrictive. Considering kernel-based regression as an example, the gradient is $(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t) - y_t)\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t) + \lambda\boldsymbol{\theta}$. Since the loss is bounded, e.g., $\|\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{x}_t) - y_t\| \leq 1$, and the RF vector in (3.8) can be bounded as $\|\mathbf{z}_{\mathbf{V}}(\mathbf{x}_t)\| \leq 1$, the constant is $L := 1 + \lambda C_\theta$ using the Cauchy-Schwartz inequality. Kernels satisfying conditions in (as3) include Gaussian, Laplacian, and Cauchy [75]. In general, (as1)-(as3) are standard in online convex optimization (OCO) [53, 89], and in kernel-based learning [75, 86, 90].

With regard to the performance of an online algorithm, static regret is commonly adopted as a metric by most OCO schemes to measure the difference between the aggregate loss of an OCO algorithm, and that of the best fixed function approximant in hindsight, e.g., [53, 89]. Specifically, for a generic sequence $\{\hat{f}_t\}$ generated by an RF-based kernel learning algorithm $\mathcal{A}$, its static regret is

$$\mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}}(T) := \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(f^*(\mathbf{x}_t)) \tag{3.25}$$

where $\hat{f}_t$ will henceforth represent $\hat{f}_t^{\mathrm{RF}}$ without the superscript for notational brevity; and, $f^*(\cdot)$ is obtained as the batch solution

$$f^*(\cdot) \in \arg \min_{\{f_p^*,\, p\in\mathcal{P}\}} \sum_{t=1}^{T} \mathcal{L}_t(f_p^*(\mathbf{x}_t)) \quad \text{with} \quad f_p^*(\cdot) \in \arg\min_{f\in\mathcal{F}_p} \sum_{t=1}^{T} \mathcal{L}_t(f(\mathbf{x}_t)) \tag{3.26}$$

with $\mathcal{F}_p := \mathcal{H}_p$, and $\mathcal{H}_p$ representing the RKHS induced by κ_p. Using (3.25) and (3.26), we first establish the static regret of our Raker approach in the following lemma.

Lemma 4: *Under (as1), (as2), and with $\hat{f}_p^*$ as in* (3.26) *with $\mathcal{F}_p := \{\hat{f}_p | \hat{f}_p(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{x}),\ \forall \boldsymbol{\theta} \in \mathbb{R}^{2D}\}$, the sequences $\{\hat{f}_{p,t}\}$ and $\{\bar{w}_{p,t}\}$ generated by Raker satisfy the following bound*

$$\sum_{t=1}^{T} \mathcal{L}_t\Big(\sum_{p=1}^{P} \bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\Big) - \sum_{t=1}^{T} \mathcal{L}_t\left(\hat{f}_p^*(\mathbf{x}_t)\right) \leq \frac{\ln P}{\eta} + \frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta} + \frac{\eta L^2 T}{2} + \eta T \tag{3.27}$$

where $\boldsymbol{\theta}_p^*$ is associated with the best RF function approximant $\hat{f}_p^*(\mathbf{x}) = \left(\boldsymbol{\theta}_p^*\right)^\top \mathbf{z}_p(\mathbf{x})$.

Proof: See Appendix 3.7.1.

Besides Raker's static regret bound, the next theorem compares the Raker loss relative to that of the best functional estimator in the original RKHS.

Theorem 1 *1 Under (as1)-(as3) and with f_p^* in* (3.26) *belonging to the RKHS $\mathcal{H}_p$, for a fixed $\epsilon > 0$, the following bound holds with probability at least* $1 - 2^8 \big(\frac{\sigma_p}{\epsilon}\big)^2 \exp\big(\frac{-D\epsilon^2}{4d+8}\big)$

$$\sum_{t=1}^{T} \mathcal{L}_t \left(\sum_{p=1}^{P} \bar{w}_{p,t} \hat{f}_{p,t}(\mathbf{x}_t) \right) - \min_{p \in \{1,\ldots,P\}} \sum_{t=1}^{T} \mathcal{L}_t \left(f_p^*(\mathbf{x}_t) \right) \leq \frac{\ln P}{\eta} + \frac{(1+\epsilon)C^2}{2\eta} + \frac{\eta L^2 T}{2} + \eta T + \epsilon LTC \tag{3.28}$$

where C is a constant, while $\sigma_p^2 := \mathbb{E}_{\mathbf{V}}^{\pi_{\kappa_p}}[\|\mathbf{v}\|^2]$ is the second-order moment of the RF vector norm. Setting $\eta = \epsilon = \mathcal{O}(1/\sqrt{T})$ in (3.28)*, the static regret in* (3.25) *leads to*

$$\text{Reg}^{\text{s}}_{\text{Raker}}(T) = \mathcal{O}(\sqrt{T}). \tag{3.29}$$

Proof: See Appendix 3.7.2.

Observe that the probability of (3.28) to hold grows as D increases, and one can always find a D to ensure a positive probability for a given ϵ. Bearing this in mind, we will henceforth use "with high probability" (w.h.p.) to summarize the sense (3.28) and (3.29) hold. Theorem 1 establishes that with proper choice of parameters, the Raker achieves sub-linear regret relative to the best static function approximant in (3.26).

3.4 Online MKL in Environments with Unknown Dynamics

Our Raker in Section 3.3 combines an ensemble of kernel learners 'on the fly,' and performs on average as the "best" fixed function, thus fulfilling the learning objective in environments with zero (or slow) dynamics. To broaden its scope to environments with unknown dynamics, this section introduces an **ada**ptive **Raker** approach (termed **AdaRaker**).

3.4.1 AdaRaker with hierarchical ensembles

As with any online learning algorithm, the choice of η in (3.19) and (3.21) affects the performance critically. Especially in environments with unknown dynamics, a large η improves the tracking ability of fast-varying functions, while a smaller one allows improved estimation of

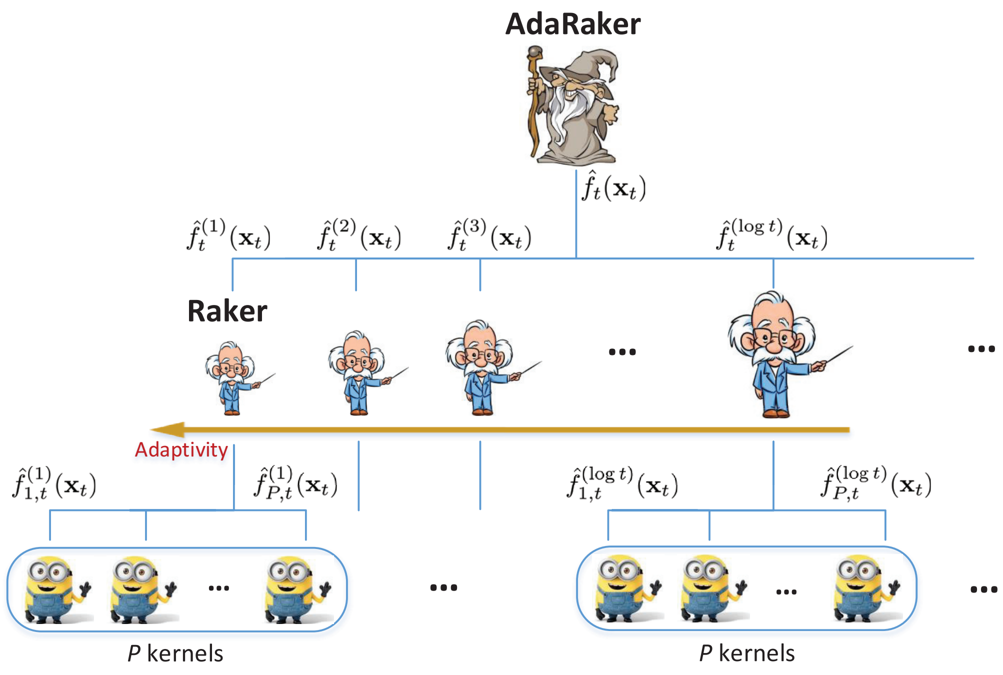

AdaRaker
$\hat{f}_t(\mathbf{x}_t)$
$\hat{f}_t^{(1)}(\mathbf{x}_t)$
$\hat{f}_t^{(2)}(\mathbf{x}_t)$
$\hat{f}_t^{(3)}(\mathbf{x}_t)$
$\hat{f}_t^{(\log t)}(\mathbf{x}_t)$
Raker
Adaptivity
$\hat{f}_{1,t}^{(1)}(\mathbf{x}_t)$
$\hat{f}_{P,t}^{(1)}(\mathbf{x}_t)$
$\hat{f}_{1,t}^{(\log t)}(\mathbf{x}_t)$
$\hat{f}_{P,t}^{(\log t)}(\mathbf{x}_t)$
P kernels
P kernels

Figure 3.1: Hierarchical AdaRaker structure. Experienced experts in the middle layer present a Raker instance, where the size of expert cartoons is proportional to the interval length.

slow-varying parameters $\{\boldsymbol{\theta}_t, w_{p,t}\}$. The optimal choice of η_t clearly depends on the variability of the optimal function estimator [68, 71]. Selecting $\{\eta_t\}$ however, is formidably challenging if the environment dynamics are unknown.

Toward addressing this challenge, our idea here is to hedge between multiple Raker learners with different learning rates. Specifically, we view each Raker instance in Algorithm 2b as a black box algorithm $\mathcal{A}_I$, where the subscript I represents the algorithm running on interval $I := [\underline{I}, \bar{I}]$ starting from slot $\underline{I}$ to slot $\bar{I}$. Let a pre-selected set $\mathcal{I}$ collect all these intervals, the design of which will be specified later. At the beginning of each interval $I \in \mathcal{I}$, a new instance of the online Raker algorithm $\mathcal{A}_I$ is initialized with an interval-specific learning rate $\eta^{(I)} := \min\{1/2, \eta_0/\sqrt{|I|}\}$ with constant $\eta_0 > 0$. Allowing for overlap between intervals, multiple Raker instances $\{\mathcal{A}_I\}$ will be run in parallel. Consider now collecting all active intervals at the current slot t in the set

$$\mathcal{I}(t) := \{I \in \mathcal{I} \,|\, t \in [\underline{I}, \bar{I}]\}, \ \forall t \in \mathcal{T}. \tag{3.30}$$

For each Raker instance $\mathcal{A}_I$ with $I \in \mathcal{I}(t)$, let $\hat{f}_t^{(I)}(\cdot)$ denote its output at time t that combines multiple kernel-based function estimators, and $\mathcal{L}_t(\hat{f}_t^{(I)}(\mathbf{x}_t))$ represent the associated

Algorithm 3b AdaRaker for online MKL in dynamic environments

```
1: **Initialization:** learner weights $\{h_1^{(I)}\}$, and their learning rates $\{\eta^{(I)}\}$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3:     Obtain $\hat{f}_t^{(I)}(\mathbf{x}_t)$ from each Raker instance $\mathcal{A}_I$, $I \in \mathcal{I}(t)$.
4:     Predict $\hat{f}_t(\mathbf{x}_t)$ via a weighted combination (3.33).
5:     Observe loss function $\mathcal{L}_t$, and incur $\mathcal{L}_t(\hat{f}_t(\mathbf{x}_t))$.
6:     **for** $I \in \mathcal{I}(t)$ **do**
7:         Incur loss $\mathcal{L}_t(\hat{f}_t^{(I)}(\mathbf{x}_t))$.
8:         Update $\hat{f}_t^{(I)}$ via Raker in Algorithm 2b.
9:         Update weights $h_{t+1}^{(I)}$ via (3.31).
10:    **end for**
11: **end for**
```

instantaneous loss. The output of the ensemble learner $\mathcal{A}$ at time t is the weighted combination of outputs from all learners, namely $\{\hat{f}_t^{(I)}, \forall I \in \mathcal{I}(t)\}$. With $h_t^{(I)}$ denoting the weight of the Raker instance $\mathcal{A}_I$, we will update it online via

$$h_{t+1}^{(I)} = \begin{cases} 0, & \text{if } t \notin I \\ \eta^{(I)}, & \text{if } t = \underline{I} \\ h_t^{(I)} \exp\left(-\eta^{(I)} r_t^{(I)}\right), & \text{else} \end{cases} \tag{3.31}$$

where $\underline{I}$ is the first time slot of interval I, and the loss of $\mathcal{A}_I$ *relative* to the overall loss is

$$r_t^{(I)} = \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \mathcal{L}_t(\hat{f}_t^{(I)}(\mathbf{x}_t)), \; \forall I \in \mathcal{I}(t). \tag{3.32}$$

Intuitively thinking, one would wish to decrease (increase) the weights of those instances with small (large) losses in future rounds. Using update (3.31), and defining the normalized weight as $\bar{h}_t^{(I)} := h_t^{(I)} / \sum_{J \in \mathcal{I}(t)} h_t^{(J)}$, the overall output is given by

$$\hat{f}_t(\mathbf{x}) := \sum_{I \in \mathcal{I}} \bar{h}_t^{(I)} \hat{f}_t^{(I)}(\mathbf{x}) \quad \text{with} \quad \hat{f}_t^{(I)}(\mathbf{x}) := \sum_{p \in \mathcal{P}} \bar{w}_{p,t}^{(I)} \hat{f}_{p,t}^{(I)}(\mathbf{x}) \tag{3.33}$$

where $\{\bar{w}_{p,t}^{(I)}\}$ are the kernel combination weights generated by Raker $\mathcal{A}_I$ (cf. (3.21)).

The AdaRaker scheme is summarized in Algorithm 3b, and depicted in Figure 3.1.

Selecting judiciously variable-length intervals in $\mathcal{I}$ can affect performance critically. Such a

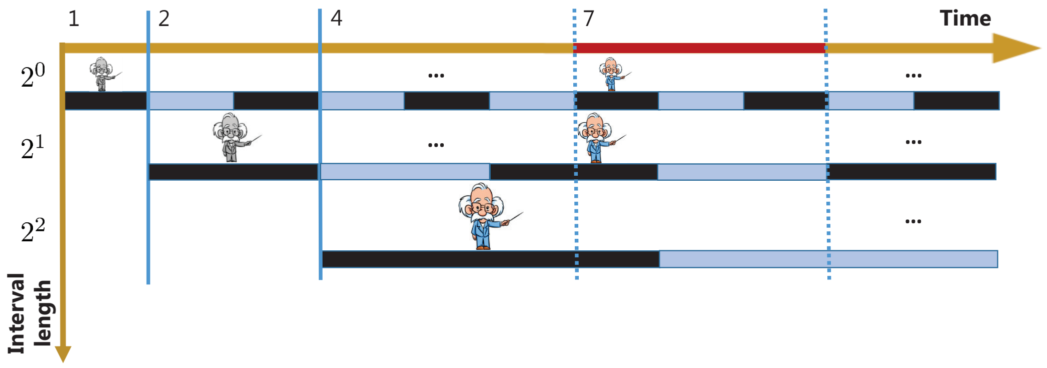


Figure 3.2: AdaRaker as an ensemble of Rakers with different learning rates: Each light/dark black interval initiates a Raker learner. At slot 7, colored experts are active, and gray ones are inactive.

selection criterion for achieving interval regret has been reported in [93]. Instead, our pursuit is a hierarchical ensemble design for online MKL in environments with unknown dynamics using scalable RF-based function approximants. This hierarchical design is well motivated because with long intervals, the Raker loss per interval is relatively low in slow-varying settings, but higher as the dynamics become more pronounced. On the other hand, a short interval can hedge against a possibly rapid change, but its performance on each interval could suffer if the objective stays nearly static. Bearing these tradeoffs in mind, we present next a simple yet efficient interval partitioning scheme.

Illustration of interval sets: *Consider partitioning the entire horizon into intervals of length* $2^0, 2^1, 2^2, \ldots$. *Intervals of length* 2^j *with a given* $j \in \mathbb{N}$ *are consecutively assigned without overlap starting from* $t = 2^j$. *In the non-overlapping case, define a set of intervals* $\mathcal{I}_j = [\underline{I}_j, \bar{I}_j]$ *such that each interval's length is* $|\mathcal{I}_j| = \bar{I}_j - \underline{I}_j + 1 = 2^j$, $j \in \mathbb{N}$. *For this selection of intervals, each time slot* t *is covered by a set of at most* $\lceil \log_2 t \rceil$ *intervals, which forms the active set of intervals* $\mathcal{I}(t)$ *at time* t. *See the diagram in Fig. 3.2.*

3.4.2 Dynamic regret analysis of AdaRaker

The static regret in Theorem 1 is with respect to a time-invariant optimal function estimator benchmark. In dynamic environments however, this optimal function benchmark may change over time - what justifies this subsection's performance analysis of AdaRaker.

Our analysis will rely on the *dynamic regret* that is related to tracking regret, and has been introduced in [71, 94] to quantify the performance of online algorithms. The dynamic regret is defined as (cf. (3.25))

$$\mathrm{Reg}^{\mathrm{d}}_{\mathcal{A}}(T) := \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(f_t^*(\mathbf{x}_t)) \tag{3.34}$$

where the benchmark is the aggregate loss incurred by a sequence of the best dynamic functions $\{f_t^*\}$ from $\mathcal{F}$ formed by the union of function spaces $\mathcal{H}_p$ induced by $\{\kappa_p\}$, given by

$$f_t^*(\cdot) \in \arg \min_{\{f_{p,t}^*,\, p\in\mathcal{P}\}} \mathcal{L}_t(f_p^*(\mathbf{x}_t)) \quad \text{with} \quad f_{p,t}^*(\cdot) \in \arg\min_{f\in\mathcal{H}_p} \mathcal{L}_t(f(\mathbf{x}_t)) \tag{3.35}$$

Comparing (3.26) with (3.35) we deduce that the dynamic regret is always larger than the static regret in (3.25). Thus, a sub-linear dynamic regret implies a sub-linear static regret, but not vice versa. Given $\{\mathcal{L}_t\}$, AdaRaker generates functions $\{\hat{f}_t\}$ to minimize the dynamic regret.

To assess the AdaRaker performance, we will start with an *intermediate* result on the static regret associated with any sub-interval $I \subseteq \mathcal{T}$.

Lemma 5: *Under (as1)-(as3), the static regret on any interval $I \subseteq \mathcal{T}$ is given by*

$$\mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}}(|I|) := \sum_{t\in I} \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \sum_{t\in I} \mathcal{L}_t(f_I^*(\mathbf{x}_t)) \tag{3.36}$$

where $|I|$ denotes the length of interval I, and the best time-invariant function approximant is $f_I^* \in \arg\min_{f\in\bigcup_{p\in\mathcal{P}}\mathcal{H}_p} \sum_{t\in I} \mathcal{L}_t(f(\mathbf{x}_t))$, with $\mathcal{H}_p$ denoting the RKHS induced by κ_p. Then for any interval $I \subseteq \mathcal{T}$ and fixed positive constants C_0, C_1, the following bound holds

$$\mathrm{Reg}^{\mathrm{s}}_{\mathrm{AdaRaker}}(|I|) \leq C_0\sqrt{|I|} + C_1 \ln T \sqrt{|I|}, \text{ w.h.p.} \tag{3.37}$$

Proof: See Appendix 3.7.3.

Lemma 5 establishes that by combining Raker learners with different learning rates, AdaRaker can achieve sub-linear static regret over *any* interval I with arbitrary interval length. This also holds for intervals overlapping with multiple intervals; see e.g., the red interval in Fig. 3.2. Clearly, the best fixed solution in (3.36) is interval specific, which can vary over different intervals. This is qualitatively why the function approximants generated by AdaRaker can cope with a *time-varying* benchmark. Such an intuition will in fact become quantitative in the next theorem, which establishes the dynamic regret for AdaRaker.

Theorem 2 *2 Suppose (as1)-(as3) are satisfied, and define the accumulated variation of online loss functions as*

$$\mathcal{V}(\{\mathcal{L}_t\}_{t=1}^T) := \sum_{t=1}^{T} \max_{f\in\mathcal{F}} \left|\mathcal{L}_{t+1}(f(\mathbf{x}_{t+1})) - \mathcal{L}_t(f(\mathbf{x}_t))\right| \tag{3.38}$$

where $\mathcal{F} := \bigcup_{p\in\mathcal{P}} \mathcal{H}_p$*. Then AdaRaker can afford a dynamic regret in* (3.34) *bounded by*

$$\begin{aligned}\mathrm{Reg}^{\mathrm{d}}_{\mathrm{AdaRaker}}(T) \leq & (2 + C_0 + C_1 \ln T) T^{\frac{2}{3}} \mathcal{V}^{\frac{1}{3}}(\{\mathcal{L}_t\}_{t=1}^T) \\ = & \tilde{\mathcal{O}}\left(T^{\frac{2}{3}} \mathcal{V}^{\frac{1}{3}}(\{\mathcal{L}_t\}_{t=1}^T)\right), \text{ w.h.p.}\end{aligned} \tag{3.39}$$

where $\tilde{\mathcal{O}}$ *neglects the lower-order terms with a polynomial* $\log T$ *rate.*

Proof: See Appendix 3.7.4.

Theorem 2 asserts that AdaRaker's dynamic regret depends on the variation of loss functions in (3.38), and on the horizon T. Interesting enough, whenever the loss functions *do not vary on average*, meaning $\mathcal{V}(\{\mathcal{L}_t\}_{t=1}^T) = \mathbf{o}(T)$, AdaRaker achieves sub-linear dynamic regret. To this end, it is useful to present an example where this argument holds.

Intermittent switches: With $\mathcal{L}_t \neq \mathcal{L}_{t+1}$ defining a switch, consider that the number of switches is sub-linear over T; that is, $\sum_{t=1}^T \mathbb{1}(\mathcal{L}_t \neq \mathcal{L}_{t+1}) = T^\gamma$, $\forall \gamma \in [0, 1)$. Then it follows that $\mathcal{V}(\{\mathcal{L}_t\}_{t=1}^T) = \mathcal{O}(T^\gamma)$, since the one-slot variation of the loss functions is bounded.

Other setups with sub-linear accumulated variation emerge, e.g., when the per-slot variation decreases as $\mathcal{V}(\mathcal{L}_t) = \mathcal{O}(t^{\gamma-1})$, $\forall\gamma \in [0, 1)$. Besides dynamic losses, sub-linear dynamic regrets can be also effected by confining the variability of optimal function estimators.

Theorem 3 *3 Suppose the conditions of Theorem 2 hold, and define the regret relative to an* m*-switching dynamic benchmark as* $\mathrm{Reg}^m_{\mathcal{A}}(T) := \sum_{t=1}^T \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \sum_{t=1}^T \mathcal{L}_t(\check{f}_t^*(\mathbf{x}_t))$*, where* $\{\check{f}_t^*\}$ *is any trajectory from*

$$\left\{\{\check{f}_t^*\}_{t=1}^T \in \bigcup_{p\in\mathcal{P}} \mathcal{H}_p \middle| \sum_{t=1}^T \mathbb{1}(\check{f}_t^* \neq \check{f}_{t-1}^*) \leq m\right\}. \tag{3.40}$$

With C_0 and C_1 denoting some universal constants, it then holds w.h.p. that

$$\mathrm{Reg}^{m}_{\mathrm{AdaRaker}}(T) \leq (C_0 + C_1 \ln T)\sqrt{Tm} = \tilde{\mathcal{O}}\left(\sqrt{Tm}\right). \tag{3.41}$$

Proof: See Appendix 3.7.5.

Theorem 3 asserts that without prior knowledge of the environment dynamics, the dynamic regret of AdaRaker is sub-linearly growing with time, provided that the number of changes of the optimal function estimators is sub-linear in T; that is, $\mathrm{Reg}^{m}_{\mathrm{AdaRaker}}(T) = \mathbf{o}(T)$ given $m = \mathbf{o}(T)$. Therefore, our AdaRaker can track the optimal dynamic functions, if the optimal function *varies slowly* over time; e.g., it does not change in the long-term average sense. While the conditions to guarantee optimality in dynamic settings may appear restrictive, they are practically relevant, since abrupt changes or adversarial samples will likely not happen at each and every slot in practice.

3.5 Numerical Tests

This section evaluates the performance of our novel algorithms in online regression tasks using both synthetic and real-world datasets.

In the subsequent tests, we use the following benchmarks.

RBF: the online *single* kernel learning method using Gaussian kernels, a.k.a. radial basis functions (RBFs), with bandwidth $\sigma^2 = \{0.1, 1, 10\}$ (cf. RBF01, RBF1, RBF10);

POLY: the online *single* kernel method using polynomial kernels, with degree $d = \{2, 3\}$ (cf. POLY2, POLY3);

LINEAR: the online *single* kernel learning method using a linear kernel;

AvgMKL: the online *single* kernel learning method using the average of candidate kernels without updating the weights;

OMKL: the popular online (O)MKL algorithm without a budget [84];

OMKL-B: the OMKL algorithm on a budget for regression modified from its single kernel version [68], with the kernel combination weights updated as (3.21);

M-Forgetron: the online multi-kernel based Forgetron modified from its single kernel version [80], with the kernel combination weights updated as in (3.21);

Time index	$[1,200]$	$[201,1000]$	$[1001,2000]$	$[2001,2300]$	$[2301,3000]$
σ^2	0.01	1	10	0.01	1
Time index	$[3001,3500]$	$[3501,4300]$	$[4301,5100]$	$[5101,5900]$	$5901,6500$
σ^2	10	0.01	1	0.01	0.1

Table 3.1: Intervals and $\{\sigma^2\}$ for synthetic dataset.

AdaMKL: the adaptive version of OMKL that operates in a similar fashion as Algorithm 3b, but instead of using our Raker as an ensemble, it adopts OMKL as an instance $\mathcal{A}_I$. Note that AdaMKL, OMKL-B, and M-Forgetron have not been formally proposed in existing works, but we introduced them here only for comparison purposes. All the considered MKL approaches use a dictionary of Gaussian kernels with $\sigma^2 = \{0.1, 1, 10\}$, and AvgMKL, OMKL, AdaMKL, OMKL-B, and M-Forgetron also include a linear, and a polynomial kernel with order of 2 into their kernel dictionary. For all MKL approaches, the stepsize for updating kernel combination weights in (3.21) is chosen as 0.5 uniformly, while the stepsize for updating per-kernel function estimators will be specified later in each test. The regularization parameter is set equal to $\lambda = 0.01$ for all approaches. Entries of $\{\mathbf{x}_t\}$ and $\{y_t\}$ are normalized to lie in $[0, 1]$. Regarding AdaMKL and AdaRaker, multiple instances are initialized on intervals with length $|I| := 2^0, 2^1, 2^2, \ldots$, along with the corresponding learning rate on the interval I as $\eta^{(I)} := \min\{1/2, 10/\sqrt{|I|}\}$; see the example in Figure 3.2. All the results in the tables were reported using the performance at the last time index.

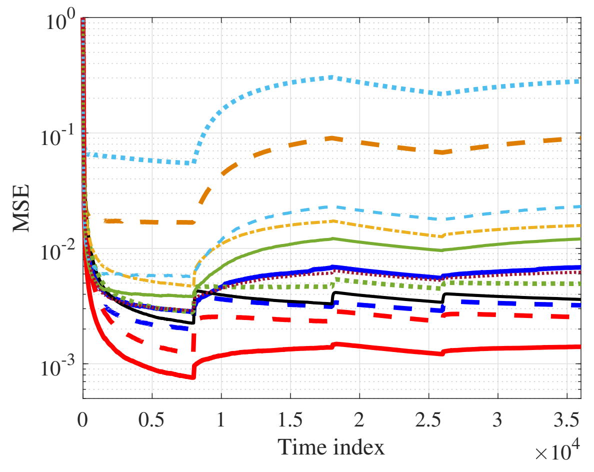
3.5.1 Synthetic data tests for regression

This subsection presents the synthetic data tests for regression.

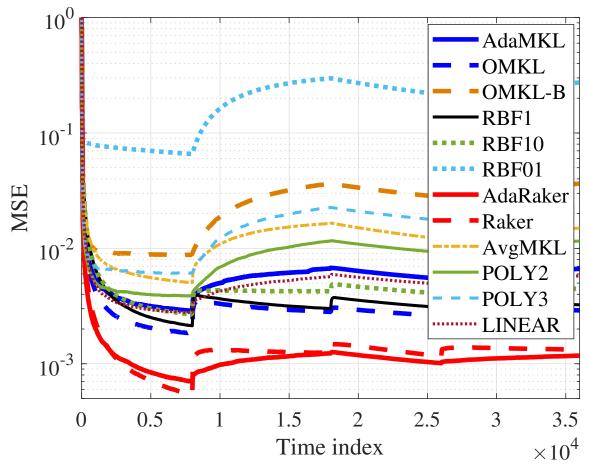
Data generation. In this test, two synthetic datasets were generated as follows.

For *Dataset 1*, the feature vectors $\{\mathbf{x}_t \in \mathbb{R}^{10}\}_{t=1}^{14,000}$ are generated from the standardized Gaussian distribution, while y_t is generated as $y_t = \sum_{\tau=1}^{t} \alpha_\tau \kappa_\tau(\mathbf{x}_t, \mathbf{x}_\tau)$, where $\{\alpha_t\}$ is generated as $\alpha_t = 1 + e_t$ with $e_t \sim \mathcal{N}(0, \sigma_\alpha^2)$ and $\sigma_\alpha = 0.01$, while $\{\kappa_t\}$ are kernel functions that change overtime: for $t \in [1, 8000] \bigcup [18001, 26000]$, κ_t is a Gaussian kernel with $\sigma^2 = 1$, while for $t \in [8001, 18000] \bigcup [26001, 36000]$ the Gaussian kernel has $\sigma^2 = 10$. Therefore, the underlying nonlinear relationship between $\mathbf{x}_t$ and y_t undergoes intermittent changes, which come from corresponding changes in the optimal kernel combinations.

Dataset 2 is generated with more variance and switching points. Specifically, the feature



(a)



(b)

Figure 3.3: MSE performance on synthetic Dataset 1: a) $D = B = 20$; b) $D = B = 50$.

vectors are generated from the standardized Gaussian distribution, while y_t is generated as $y_t = \sum_{\tau=1}^{t} \alpha_\tau \kappa_\tau(\mathbf{x}_t, \mathbf{x}_\tau)$, where $\{\kappa_t\}$ change over 10 intervals with different σ^2; see Table 3.1.

Testing performance. The performance of all schemes is tested in terms of the mean-square (prediction) error $\text{MSE}(t) := (1/t)\sum_{\tau=1}^{t}(y_\tau - \hat{y}_\tau)^2$ in Figure 3.3 and Figure 3.4, and their CPU time is listed in Table 3.2. For OMKL-B, $B = 20$ and 50 most recent data samples were kept in the budget; and for RF-based Raker and AdaRaker approaches, $D = 20$ and 50 orthogonal random features were used by default. The default stepsize is chosen as $1/\sqrt{T}$ for RBF, POLY, LINEAR, AvgMKL, OMKL, OMKL-B and Raker. In both tests, AdaRaker outperforms the alternatives in terms of MSE, especially when the true nonlinear relationship between $\mathbf{x}_t$ and y_t changes; e.g., compare the MSE of KL-RBF and Raker with that of AdaRaker at $t = 8000, 18000, 26000$ in Figure 3.3, and $t = 200, 2000, 3000, 3500$ in Figure 3.4. This corroborates the effectiveness of the novel AdaRacker method that can flexibly select learning rates according to the variability of the environments, and adaptively combine multiple kernels when the optimal underlying nonlinear relationship is varying over time. In addition, MKL approaches including our Raker approach enjoy lower MSE than that of the single-kernel approaches as well as the simple AvgMKL approach, which is also aligned with our design principle of developing MKL schemes that broaden generalizability of a kernel-based learner over a larger function space.

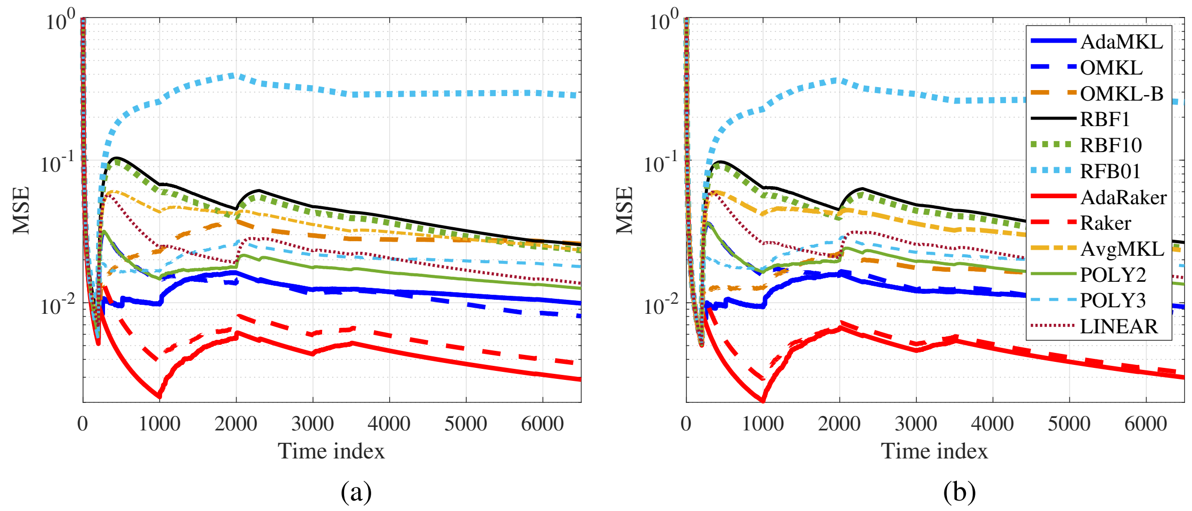


Figure 3.4: MSE performance on synthetic Dataset 2: a) $D = B = 20$; b) $D = B = 50$.

Table 3.2 records the CPU time of all benchmark algorithms running tests on two different datasets. It can be observed that leveraging the RF-based approximation, the proposed AdaRaker and Raker algorithms are much faster than AdaMKL and OMKL; hence, they are preferable especially for large-scale datasets. Although the CPU time of OMKL-B with a budget size $B = 20$ or $B = 50$ is relatively low, OMKL-B does not perform as well as AdaRaker and Raker algorithms. Therefore, the AdaRaker and Raker approaches attain a sweet-spot in the performance-complexity tradeoff.

3.5.2 Real data tests for online regression

To further evaluate our algorithms in real-world scenarios, the present subsection is devoted to testing and comparing on several popular real datasets.

Datasets description. Performance is tested on benchmark datasets from UCI machine learning repository [95].

- **Twitter** dataset consists of $T = 14,000$ samples from a popular micro-blogging platform Twitter, where $\mathbf{x}_t \in \mathbb{R}^{77}$ include features such as the number of new interactive authors, and the length of discussion on a given topic, while y_t represents the average number of active discussion (popularity) on a certain topic [96]. A larger dataset with $T = 100,000$ is also included for testing only (Ada)Raker and OMKL-B, since other methods do not scale to such a large T.

	Dataset 1		**Dataset 2**	
Setting	$D = B = 20$	$D = B = 50$	$D = B = 20$	$D = B = 50$
AdaMKL	318.52		27.29	
OMKL	157.10		5.47	
RBF	47.83		1.06	
POLY2	6.01		0.47	
POLY3	28.27		1.24	
LINEAR	4.80		0.35	
AvgMKL	144.85		5.02	
OMKL-B	3.75	4.05	0.72	0.77
Raker	1.39	1.53	0.18	0.20
AdaRaker	21.94	24.24	3.32	3.54

Table 3.2: CPU time (in seconds) on synthetic datasets. RBF, POLY represents all single-kernel methods using RBF and polynomial kernels, since they have the same CPU time.

Dataset	**# features** (d)	**# samples** (T)	**feature type**
Twitter	77	$14,000$	real & integer
Twitter (Large)	77	$100,000$	real & integer
Tom's hardware	96	$10,000$	real & integer
Energy	27	$18,600$	real
Air quality	13	$9,358$	real

Table 3.3: A summary of real datasets used in the tests.

- **Tom's hardware** dataset contains $T = 10,000$ samples from a worldwide new technology forum, where a 96-dimensional feature vector includes the number of discussions involving a certain topic, while y_t represents the average number of display about a certain topic on Tom's hardware [96].

- **energy** dataset consists of $T = 18,600$ samples, with each $\mathbf{x}_t \in \mathbb{R}^{27}$ describing the humidity and temperature indoors and outdoors, while y_t denotes the energy use of light fixtures in the house [97].

- **air quality** dataset collects $T = 9,358$ instances of hourly averaged responses from five chemical sensors located in a polluted area of Italy. The averaged sensor response $\mathbf{x}_t \in \mathbb{R}^{13}$ contains the hourly concentrations of e.g., CO, Non Metanic Hydrocarbons, and Nitrogen Dioxide (NO2), where the goal is to predict the concentration of polluting chemicals y_t in the air [98].

Algorithms/ Datasets	Twitter	Tom's	Energy	Air
RBF ($\sigma^2 = 0.1$)	27.0	14.4	28.9	26.3
RBF ($\sigma^2 = 1$)	13.5	17.0	28.8	12.7
RBF ($\sigma^2 = 10$)	23.3	18.8	28.8	15.5
POLY2	12.7	22.3	28.8	7.34
POLY3	20.4	22.7	28.9	5.91
LINEAR	8.57	19.5	28.8	10.7
AvgMKL	14.4	17.5	28.7	11.9
OMKL	8.55	14.3	28.1	6.4
AdaMKL	16.1	18.4	30.4	10.1
OMKL-B ($B = 50$)	27.0	22.1	73.3	35.9
Raker ($D = 50$)	3.0	3.4	19.3	2.0
AdaRaker ($D = 50$)	**2.6**	**1.9**	**13.8**	**1.3**

Table 3.4: MSE (10^{-3}) performance of different algorithms with stepsize $1/\sqrt{T}$.

To highlight the effectiveness of our approaches, the datasets mainly include time series data, where non-stationarity is more likely to happen; see Table 4.1 for a summary.

MSE performance. The MSE performance of each algorithm on the aforementioned datasets is presented in Table 3.4. By default, we use the complexity $B = D = 50$ for OMKL-B and (Ada)Raker, and the stepsize $1/\sqrt{T}$ for RBF, POLY, LINEAR, AvgMKL, OMKL, OMKL-B and Raker. To boost the performance of each algorithm, their MSE when using manually tuned stepsizes is also reported in Table 3.5, which selects the best stepsize on each dataset among $\{10^{-3}, 10^{-2}, \cdots, 10^{3}\}/\sqrt{T}$. A common observation is that leveraging the flexibility of multiple kernels, MKL methods in most cases outperform the algorithms using only a single kernel. By simply averaging over all the kernels, AvgMKL outperforms most of single kernel methods, but performs worse than the adaptive kernel combination methods. This confirms that relying on a pre-selected kernel function is not sufficient to guarantee low fitting loss, while allowing the MKL approaches to select the best kernel combinations in a data-driven fashion holds the key for improved performance.

In most tested datasets, Raker obtains function approximants with lower MSE relative to MKL alternatives without RF approximation. Furthermore, incorporating multiple Raker instances with variable learning rates, AdaRaker consistently yields the lowest MSE in all the tests. As it has been shown in the synthetic data test, the sizable performance gain of AdaRaker appears when the underlying nonlinear models change in the tested time-series datasets. This

Algorithms/ Datasets	Twitter	Tom's	Energy	Air
RBF ($\sigma^2 = 0.1$)	17.2	3.3	16.6	8.1
RBF ($\sigma^2 = 1$)	3.3	5.1	16.4	2.8
RBF ($\sigma^2 = 10$)	5.6	13.6	16.4	18.9
POLY2	8.1	15.9	16.2	3.3
POLY3	20.4	20.7	16.2	4.6
LINEAR	2.7	4.8	16.3	2.9
AvgMKL	7.1	6.2	16.3	2.8
OMKL	4.2	3.3	16.2	2.4
AdaMKL	16.1	18.4	30.4	10.1
OMKL-B ($B = 50$)	9.9	11.8	19	7.1
Raker ($D = 50$)	2.9	2.6	**13.8**	**1.3**
AdaRaker ($D = 50$)	**2.6**	**1.9**	**13.8**	**1.3**

Table 3.5: MSE (10^{-3}) performance of different algorithms with optimally chosen stepsizes.

MSE	**OMKL-B**				**Raker**				**AdaRaker**
Stepsize	$1/\sqrt{T}$	$0.5/\sqrt{t}$	$0.1/\sqrt{t}$	Tuned	$1/\sqrt{T}$	$0.5/\sqrt{t}$	$0.1/\sqrt{t}$	Tuned	/
Twitter	27.0	27.1	29.6	9.9	3.0	17.9	4.3	2.9	**2.6**
Tom's	22.1	22.1	22.6	11.8	3.4	2.0	7.6	2.6	**1.9**
Energy	73.3	74.1	79.5	19.0	19.3	29.5	25.1	**13.8**	**13.8**
Air	35.9	35.9	40.1	7.1	2.0	29.1	4.0	**1.3**	**1.3**
Twitter (Large)	20.7	27.2	28.0	11.3	3.2	3.1	3.3	3.0	**2.7**

Table 3.6: MSE (10^{-3}) versus the choice of stepsizes with complexity $B = D = 50$.

observation is aligned with our design principle of AdaRaker; that is, when the optimal function predictor varies slowly (fast), AdaRaker tends to select a Raker instance with small (large) learning rate. Interesting enough, even with adaptive learning rate, AdaMKL does not perform as well as OMKL in some tests. This is partially because unlike AdaRaker with fixed number of RFs, each instance in AdaMKL involves a different *number of support vectors* (samples). The instance operating on the longest interval contains at most $T/2$ support vectors, which may deteriorate performance relative to OMKL with T support vectors.

Table 4.2 further compares the MSE performance of AdaRaker with OMKL-B and Raker using different stepsizes. Clearly, the performance of OMKL-B and Raker is sensitive to the choice of stepsizes. While the optimal stepsize varies from dataset to dataset, selecting a constant stepsize $1/\sqrt{T}$ generally leads to better performance than a diminishing one of $\mathcal{O}(1/\sqrt{t})$. In the online scenarios however, the choice $1/\sqrt{T}$ may not be feasible if T is unknown ahead of time. In contrast, AdaRaker obtained the best MSE performance without knowing T, and

Algorithms/ Datasets	Twitter	Tom's	Energy	Air
RBF	54.7	32.5	26.6	1.71
POLY2	2.25	1.16	2.42	0.58
POLY3	5.62	2.97	7.90	1.51
LINEAR	1.83	0.98	196	0.39
AvgMKL	148.4	81.6	82.4	5.29
OMKL	153.5	81.9	83.3	5.90
AdaMKL	164.1	102.7	117.9	35.3
OMKL-B ($B = 50$)	1.89	1.42	2.02	0.89
Raker ($D = 50$)	$\mathbf{0.51}$	$\mathbf{0.38}$	$\mathbf{0.65}$	$\mathbf{0.28}$
AdaRaker ($D = 50$)	8.64	6.03	10.94	5.28

Table 3.7: A summary of CPU time (second) on real datasets.

without the need of stepsize selection, which confirms that AdaRaker is capable of adapting its stepsize to variable environments with unknown dynamics.

Computational complexity. The CPU time of all the considered schemes is recorded under all the tests; see Table 3.7. It is evident that in all tests, our RF-based MKL methods including Raker and AdaRaker are computationally more efficient than other MKL methods except that OMKL-B is faster than AdaRaker. Intuitively speaking, the per-slot complexity of Raker does not grow with time, since it requires computing only one inner product of two $2D$-dimensional vectors per kernel learner, while the computational complexity of AdaMKL, OMKL, POLY, LINEAR, AvgMKL, and RBF increases with time at least linearly. With a fixed budget size, OMKL-B enjoys light-weight updates that leads to a lower CPU time than alternatives, but higher than Raker. However, given such a limited budget of data, OMKL-B exhibits higher MSE than AdaRaker and Raker; see MSE in Tables 3.4 and 4.2.

Running multiple instances of Raker in parallel, the complexity of AdaRaker is reasonably higher than Raker (roughly $\log T$ times higher), but its runtime is still only around 10% of that of AdaMKL, and significantly lower than other single-kernel alternatives especially when the actual feature dimension d is higher than the number of random features D. The computational advantage of our MKL algorithms in this test also corroborates the quantitative analysis at the end of Section 3.3.2. Regarding the tradeoff between learning accuracy and complexity, a delicate comparison among OMKL-B, Raker and AdaRaker follows next.

Accuracy versus complexity. To further understand the tradeoff between complexity and learning accuracy, the performance of three scalable methods AdaRaker, Raker and OMKL-B is

MSE	**OMKL-B**			**Raker**			**AdaRaker**		
Complexity	10	50	100	10	50	100	10	50	100
Twitter	28.9	27.0	26.1	5.9	3.0	3.0	3.8	**2.6**	**2.6**
Tom's	22.7	22.1	21.7	8.1	3.4	2.3	7.0	1.9	**1.8**
Energy	79.1	73.3	67.9	25.7	19.3	16.4	18.7	13.8	**13.3**
Air pollution	36.7	35.9	35.8	10.1	2.0	1.7	4.3	1.3	**1.2**
Twitter (Large)	25.0	20.7	19.0	3.9	3.2	3.0	3.3	**2.7**	**2.7**

Table 3.8: MSE (10^{-3}) versus complexity. For OMKL-B, the complexity measure is the data budget B; and for (Ada)Raker, the complexity measure is the number of RFs D.

Time	**OMKL-B**			**Raker**			**AdaRaker**		
Complexity	10	50	100	10	50	100	10	50	100
Twitter	1.42	1.89	2.84	0.42	0.51	0.80	7.58	8.64	11.65
Tom's	1.00	1.42	2.81	0.41	0.38	0.56	5.09	6.03	8.98
Energy	1.84	2.02	2.32	0.58	0.65	0.76	9.96	10.94	12.47
Air pollution	0.82	0.89	0.97	0.24	0.28	0.32	4.09	5.28	5.29
Twitter (Large)	12.90	16.34	23.6	6.07	6.63	8.42	67.10	78.10	109.40

Table 3.9: CPU time (second) versus complexity of B for OMKL-B, and D for (Ada)Raker.

tested under different parameter settings, e.g., D, the number of random features, and B, the number of budgeted data. The MSE performance is reported in Table 3.8 after one pass of all data in each dataset, while the corresponding CPU time is in Table 3.9.

Not surprising, all three algorithms require longer CPU time as the complexity (in terms of B or D) increases. For given complexity (same B and D), Raker requires the lowest CPU time, and its MSE is also markedly lower than that of OMKL-B in all tests. On the other hand, AdaRaker always attains the lowest MSE, and its performance gain is remarkable especially in the Energy and Air pollution datasets. For Twitter (Large) dataset, the performance of AdaRaker does not improve as RFs increase from $D = 50$ to $D = 100$, which implies that $D = 50$ is enough to provide reliable kernel approximation in this dataset. Considering that AdaRaker is embedded with concurrent $\log t$ Raker instances at time t, its CPU time is relatively higher. However, one would expect a major reduction in the number of concurrent instances and thus markedly lower CPU time, if a larger basic interval size (instead of base number 2 in Figure 3.2) is incorporated in AdaRaker real implementation.

	Classification error			CPU time		
Algorithms/Datasets	Movement	Devices	Activity	Movement	Devices	Activity
RBF ($\sigma^2 = 0.1$)	43.1	6.67	0.46	2.76	2.13	4.42
RBF ($\sigma^2 = 1$)	41.3	28.1	5.21	2.79	2.04	4.42
RBF ($\sigma^2 = 10$)	40.3	31.8	41.1	2.62	2.09	4.43
POLY2	43.5	14.3	3.13	1.63	0.31	0.81
POLY3	43.6	25.2	2.39	4.75	0.57	1.79
LINEAR	43.8	47.4	4.30	1.26	0.21	0.60
AvgMKL	41.7	23.9	3.16	10.2	6.72	14.67
OMKL	38.2	16.0	0.60	10.27	7.07	14.46
AdaMKL	3.5	0.86	0.98	33.77	10.51	21.46
M-Forgetron ($B = 50$)	1.64	0.53	1.14	0.92	0.27	0.53
Raker ($D = 50$)	9.74	2.54	0.58	**0.40**	**0.12**	**0.21**
AdaRaker ($D = 50$)	**1.10**	**0.36**	**0.34**	6.76	1.73	3.56

Table 3.10: Classification error ($\%$) and runtime (second) of different algorithms with the default stepsize $1/\sqrt{T}$ for RBF, OMKL and Raker, and with complexity $B = D = 50$.

At this point, one may wonder how many RFs are enough for Raker and AdaRaker to guarantee the same online learning accuracy as that of OMKL-B with B samples. While this intriguing question has been recently studied in the batch setting with an answer of $D = \mathcal{O}(\sqrt{B})$ RFs [99], its thorough treatment in the online setting constitutes our future research.

3.5.3 Real data tests for online classification

In this section, the performance of Raker and AdaRaker is tested on real datasets for the online classification task. We use the logistic loss as the learning objective function with the regularization parameter $\lambda = 0.005$ for all considered approaches except for the perceptron-based Forgetron algorithm. Kernels and all other parameters such as the default stepsizes, are chosen as those in the regression task.

Datasets description. We test classification performance on the following datasets.

- **Movement** dataset consists of $T = 13,197$ temporal streams of received signal strength (RSS) measured between the nodes of a wireless sensor network, with each $\mathbf{x}_t \in \mathbb{R}^4$ comprising 4 anchor nodes [100]. Data has been collected during user movements at the frequency of 8 Hz (8 samples per second). The RSS samples in the dataset have been rescaled to lie in $[-1, 1]$. The binary label y_t indicates whether the user's trajectory will lead to a change in the spatial context (here a room change) or not.

	Classification error		
Algorithms/Datasets	Movement	Devices	Activity
RBF ($\sigma^2 = 0.1$)	28.9	5.16	0.26
RBF ($\sigma^2 = 1$)	1.27	0.42	0.53
RBF ($\sigma^2 = 10$)	**1.10**	0.36	1.14
POLY2	8.19	1.7	0.56
POLY3	15.2	17.3	0.45
LINEAR	7.46	30.7	0.60
AvgMKL	1.69	2.26	0.48
OMKL	**1.10**	0.36	0.29
AdaMKL	3.50	0.86	1.00
M-Forgetron ($B = 50$)	1.64	0.53	1.14
Raker ($D = 50$)	**1.10**	**0.28**	**0.24**
AdaRaker ($D = 50$)	**1.10**	0.36	0.34

Table 3.11: Classification error (%) of different algorithms with the dataset-specific optimally chosen stepsizes for RBF, OMKL and Raker, and with complexity $B = D = 50$.

	OMKL				**Raker**				**AdaRaker**
Stepsize	$1/\sqrt{T}$	$1/\sqrt{t}$	$10/\sqrt{t}$	Tuned	$1/\sqrt{T}$	$1/\sqrt{t}$	$10/\sqrt{t}$	Tuned	/
Movement	38.2	39.5	22.3	1.10	12.1	8.60	1.79	1.10	**1.10**
Devices	16.0	13.2	6.06	0.36	2.54	2.04	0.53	**0.28**	0.36
Activity	0.60	0.53	0.50	0.29	0.58	0.52	0.54	**0.24**	0.34

Table 3.12: Classification error (%) versus different choices of stepsizes with $B = D = 50$.

- **Electronic Device** dataset consists of $T = 3,600$ samples collected as part of a government sponsored study called 'Powering the Nation,' where the feature vectors $\mathbf{x}_t \in \mathbb{R}^{60}$ represent electricity readings from different households over 15 mins, sampled within a month [101]. Binary label y_t represents the type of electronic devices used at the certain interval of time time: dishwasher or kettle.
- **Human Activity** dataset consists of $T = 7,352$ samples collected from a group of 30 volunteers wearing a smartphone (Samsung Galaxy S II) on their waist to monitor activities [102]. Feature vectors $\{\mathbf{x}_t \in \mathbb{R}^{30}\}$ here measure e.g., triaxial acceleration and angular velocity, while binary label y_t represents the activity during a certain period: walking or not walking.

Classification performance. The classification error $(1/T)\sum_{t=1}^{T} \max\{0, \text{sign}(-y_t\hat{y}_t)\}$ and the CPU time of each algorithm on these datasets are summarized in Table 3.10 when a default

	Classification error			**CPU time**		
Algorithms/ Datasets	Movement	Devices	Activity	Movement	Devices	Activity
M-Forgetron ($B = 10$)	1.60	0.53	1.14	0.92	0.26	0.53
M-Forgetron ($B = 50$)	1.64	0.53	1.14	0.92	0.27	0.53
M-Forgetron ($B = 100$)	1.42	0.53	1.14	0.94	0.29	0.53
Raker ($D = 10$)	26.3	8.37	3.26	**0.35**	**0.10**	**0.18**
Raker ($D = 50$)	9.74	2.54	0.58	0.40	0.12	0.21
Raker ($D = 100$)	4.65	1.53	0.43	0.46	0.15	0.26
AdaRaker ($D = 10$)	2.46	0.66	0.68	6.13	0.65	3.22
AdaRaker ($D = 50$)	**1.10**	**0.36**	**0.34**	6.76	1.73	3.56
AdaRaker ($D = 100$)	**1.10**	**0.36**	**0.34**	7.55	2.04	4.22

Table 3.13: Classification error (%) and CPU time (second) versus complexity.

stepsize $1/\sqrt{T}$ is used for POLY, LINEAR, RBF, AvgMKL, OMKL and Raker. The budget of M-Forgetron is set at $B = 50$ samples, while Raker and AdaRaker adopt $D = 50$ RFs. As with the regression tests, it is evident that AdaRaker attains the highest classification accuracy and the Raker has the lowest CPU time among all competing algorithms. Without having to tune stepsizes, the performance of AdaMKL and M-Forgetron is also competitive in this case. To explore the best performance of each algorithm, the classification performance under manually tuned stepsizes is reported in Table 3.11, where each algorithm uses the best stepsize among $\{10^{-3}, 10^{-2}, \cdots, 10^{3}\}/\sqrt{T}$ for each dataset. With the optimally chosen stepsizes, the performance of all algorithms improves, and Raker even achieves slightly lower classification error than AdaRaker in some datasets. This is reasonable since compared to Raker with the offline tuned stepsize, AdaRaker will incur some error due to the online adaptation to several (possibly suboptimal) learning rates.

To corroborate the effectiveness of our algorithms in adapting to unknown dynamics (e.g., unknown time horizon T and variability), Table 3.12 compares the performance of AdaRaker with OMKL and Raker using default, diminishing and optimally tuned stepsizes. Similar to regression tests, the performance of Raker and OMKL is sensitive to the stepsize choice, while AdaRaker achieves the desired performance by combining learners with different learning rates. By simply averaging over all the kernels, AvgMKL outperforms single kernel methods in most cases, but performs much worse than OMKL and (Ada)Raker methods. Note that the Raker also achieves competitive classification accuracy when the constant stepsize $1/\sqrt{T}$ is used. Such a choice is however not always feasible in practice, since it requires knowledge of how many data samples will be available ahead of time.

Accuracy versus complexity. In this experiment, we test classification performance in terms of both classification error and CPU time for different levels of complexity; see Table 3.13. We use the number of support vectors B for M-Forgetron, and the number of RFs D for Raker and AdaRaker to represent different levels of complexity, and compare their performance using the default stepsize. It is expected that CPU time increases as the complexity increases, and the classification error decreases as the complexity grows. For all three datasets, the AdaRaker achieves the lowest classification error, and the Raker outperforms the M-Forgetron while at the same time it is more efficient computationally.

3.6 Summary

This chapter dealt with kernel-based learning in environments with unknown dynamics that also include static or slow variations. Uniquely combining advances in random feature based function approximation with online learning from an ensemble of experts, a scalable online multi-kernel learning approach termed Raker, was developed for static environments based on a dictionary of kernels. Endowing Raker with capability of tracking time-varying optimal function estimators, AdaRaker was introduced as an ensemble version of Raker with variable learning rates. The key modules of the novel learning approaches are: i) the random features are for scalability, as they reduce the per-iteration complexity; ii) the preselected kernel dictionary is for flexibility, that is to broaden generalizability of a kernel-based learner over a larger function space; iii) the weighted combination of kernels adjusted online accounts for the reliability of learners; and, iv) the adoption of multiple learning rates is for improved adaptivity to changing environments with unknown dynamics.

Complementing the principled algorithmic design, the performance of Raker is rigorously established using static regret analysis. Furthermore, without a-priori knowledge of dynamics, it is proved that AdaRaker achieves sub-linear dynamic regret, provided that either the loss or the optimal learning function does not change on average. Experiments on synthetic and real datasets validate the effectiveness of the novel methods.

3.7 Appendix

3.7.1 Proof of Lemma 4

To prove Lemma 4, we introduce two intermediate lemmata as follows.

Lemma 6: *Under (as1), (as2), and $\hat{f}_p^*$ as in* (3.26) *with $\mathcal{F}_p := \{\hat{f}_p | \hat{f}_p(\mathbf{x}) = \boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{x}), \forall \boldsymbol{\theta} \in \mathbb{R}^{2D}\}$, let $\{\hat{f}_{p,t}(\mathbf{x}_t)\}$ denote the sequence of estimates generated by Raker with a pre-selected kernel κ_p. Then the following bound holds true w.p.1*

$$\sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_{p,t}(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_p^*(\mathbf{x}_t)) \leq \frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta} + \frac{\eta L^2 T}{2} \tag{3.42}$$

where η is the learning rate, L is the Lipschitz constant in (as2), and $\boldsymbol{\theta}_p^*$ is the corresponding parameter (or weight) vector supporting the best estimator $\hat{f}_p^*(\mathbf{x}) = (\boldsymbol{\theta}_p^*)^\top \mathbf{z}_p(\mathbf{x})$.

Proof: Similar to the regret analysis of online gradient descent [53], using (3.12) for any fixed $\boldsymbol{\theta}$, we find

$$\begin{aligned} \|\boldsymbol{\theta}_{p,t+1} - \boldsymbol{\theta}\|^2 =& \|\boldsymbol{\theta}_{p,t} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t) - \boldsymbol{\theta}\|^2 \\ =& \|\boldsymbol{\theta}_{p,t} - \boldsymbol{\theta}\|^2 + \eta^2 \|\nabla \mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t)\|^2 - 2\eta \nabla^\top \mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t)(\boldsymbol{\theta}_{p,t} - \boldsymbol{\theta}). \end{aligned} \tag{3.43}$$

Meanwhile, the convexity of the loss under (as1) implies that

$$\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t) - \mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{x}_t), y_t) \leq \nabla^\top \mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t)(\boldsymbol{\theta}_{p,t} - \boldsymbol{\theta}). \tag{3.44}$$

Plugging (3.44) into (3.43) and rearranging terms yields

$$\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t) - \mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{x}_t), y_t) \leq \frac{\|\boldsymbol{\theta}_{p,t} - \boldsymbol{\theta}\|^2 - \|\boldsymbol{\theta}_{p,t+1} - \boldsymbol{\theta}\|^2}{2\eta} + \frac{\eta}{2} \|\nabla \mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t)\|^2. \tag{3.45}$$

Summing (3.45) over $t = 1, \ldots, T$, with $\hat{f}_{p,t}(\mathbf{x}_t) = \boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t)$, we arrive at

$$\begin{aligned} &\sum_{t=1}^{T} \Big(\mathcal{L}(\hat{f}_{p,t}(\mathbf{x}_t), y_t) - \mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{x}_t), y_t) \Big) \\ \leq& \frac{\|\boldsymbol{\theta}_{p,1} - \boldsymbol{\theta}\|^2 - \|\boldsymbol{\theta}_{p,T+1} - \boldsymbol{\theta}\|^2}{2\eta} + \frac{\eta}{2} \sum_{t=1}^{T} \|\nabla \mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{x}_t), y_t)\|^2 \end{aligned}$$

$$\overset{(a)}{\leq} \frac{\|\boldsymbol{\theta}\|^2}{2\eta} + \frac{\eta L^2 T}{2} \tag{3.46}$$

where (a) uses the Lipschitz constant in (as2), the non-negativity of $\|\boldsymbol{\theta}_{p,T+1} - \boldsymbol{\theta}\|^2$, and the initial value $\boldsymbol{\theta}_{p,1} = \mathbf{0}$. The proof of Lemma 6 is then complete by choosing $\boldsymbol{\theta} = \boldsymbol{\theta}_p^* = \sum_{t=1}^{T} \alpha_{p,t}^* \mathbf{z}_p(\mathbf{x}_t)$ such that $\hat{f}_p^*(\mathbf{x}_t) = \boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{x}_t)$ in (3.46). Lemma 6 establishes that the static regret of the Raker is upper bounded by some constants, which mainly depend on the stepsize in (3.19) and the time horizon T.

In addition, we will bound the difference between the loss of the solution obtained from Algorithm 2b and the loss of the best single kernel-based online learning algorithm. Specifically the following lemma holds:

Lemma 7: *Under (as1) and (as2), with $\{\hat{f}_{p,t}\}$ generated from Raker, it holds that*

$$\sum_{t=1}^{T}\sum_{p=1}^{P} \bar{w}_{p,t}\mathcal{L}_t(\hat{f}_{p,t}(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_{p,t}(\mathbf{x}_t)) \leq \eta T + \frac{\ln P}{\eta} \tag{3.47}$$

where η is the learning rate in (3.21)*, and P is the number of kernels in the dictionary.*

Proof: Letting $W_t := \sum_{p=1}^{P} w_{p,t}$, the weight recursion in (3.21) implies that

$$\begin{aligned} W_{t+1} &= \sum_{p=1}^{P} w_{p,t+1} = \sum_{p=1}^{P} w_{p,t} \exp\left(-\eta \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)\right) \\ &\leq \sum_{p=1}^{P} w_{p,t}\left(1 - \eta\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta^2 \mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2\right) \end{aligned} \tag{3.48}$$

where the last inequality holds because $\exp(-\eta x) \leq 1 - \eta x + \eta^2 x^2$, for $|\eta| \leq 1$. Furthermore, substituting $\bar{w}_{p,t} := w_{p,t}/\sum_{p=1}^{P} w_{p,t} = w_{p,t}/W_t$ into (3.48), it follows that

$$\begin{aligned} W_{t+1} &\leq \sum_{p=1}^{P} W_t \bar{w}_{p,t}\left(1 - \eta\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta^2\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2\right) \\ &= W_t\left(1 - \eta\sum_{p=1}^{P} \bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta^2\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right)^2\right). \end{aligned} \tag{3.49}$$

Using $1 + x \leq e^x$, $\forall x$, (3.49) leads to

$$W_{t+1} \leq W_t \exp\Bigg(- \eta \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right) + \eta^2 \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right)^2 \Bigg). \tag{3.50}$$

Telescoping (3.50) from $t = 1$ to T, we have ($W_1 = 1$)

$$W_{T+1} \leq \exp\Bigg(- \eta \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right) + \eta^2 \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right)^2 \Bigg). \tag{3.51}$$

On the other hand, for any p, the following holds true

$$\begin{aligned} W_{T+1} \geq w_{p,T+1} &= w_{p,1} \prod_{t=1}^{T} \exp(-\eta \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right)) \\ &= w_{p,1} \exp\Bigg(- \eta \sum_{t=1}^{T} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right) \Bigg). \end{aligned} \tag{3.52}$$

Combining (3.51) with (3.52), we arrive at

$$\begin{aligned} &\exp\Bigg(-\eta \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right) + \eta^2 \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right)^2 \Bigg) \\ &\geq w_{p,1} \exp\Bigg(-\eta \sum_{t=1}^{T} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right) \Bigg). \end{aligned} \tag{3.53}$$

Taking the logarithm on both sides of (3.53), we find that (cf. $w_{p,1} = 1/P$)

$$-\eta \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t \Big(\hat{f}_{p,t}(\mathbf{x}_t) \Big) + \eta^2 \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t \Big(\hat{f}_{p,t}(\mathbf{x}_t) \Big)^2 \geq -\eta \sum_{t=1}^{T} \mathcal{L}_t \Big(\hat{f}_{p,t}(\mathbf{x}_t) \Big) - \ln P \tag{3.54}$$

which leads to

$$\sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right) \leq \sum_{t=1}^{T} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right) + \eta \sum_{t=1}^{T} \sum_{p=1}^{P} \bar{w}_{p,t} \mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right)^2 + \frac{\ln P}{\eta} \tag{3.55}$$

and the proof is complete since $\mathcal{L}_t \left(\hat{f}_{p,t}(\mathbf{x}_t) \right)^2 \leq 1$ and $\sum_{p=1}^{P} \bar{w}_{p,t} = 1$.

Moreover, since $\mathcal{L}_t(\cdot)$ is convex under (as1), Jensen's inequality implies that

$$\mathcal{L}_t\bigg(\sum_{p=1}^{P}\bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\bigg) \leq \sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right). \tag{3.56}$$

Plugging (3.56) into (3.47) in Lemma 7, we arrive at

$$\begin{aligned}\sum_{t=1}^{T}\mathcal{L}_t\bigg(\sum_{p=1}^{P}\bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\bigg) &\leq \sum_{t=1}^{T}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{x}_t)\right) + \eta T + \frac{\ln P}{\eta}\\ &\overset{(a)}{\leq} \sum_{t=1}^{T}\mathcal{L}_t\left(\hat{f}_p^*(\mathbf{x}_t)\right) + \frac{\ln P}{\eta} + \frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta} + \frac{\eta L^2 T}{2} + \eta T \end{aligned}\tag{3.57}$$

where (a) follows because $\boldsymbol{\theta}_p^*$ is the optimal solution for any given kernel κ_p. This proves the claim in Lemma 4.

3.7.2 Proof of Theorem 1

To derive the performance bound relative to the best function estimator $f^*(\mathbf{x}_t)$ in the RKHS, the key step is to bound the error of approximation. For a given shift-invariant κ_p, the maximum point-wise error of the RF kernel approximant is uniformly bounded with probability at least $1 - 2^8\left(\frac{\sigma_p}{\epsilon}\right)^2\exp\left(\frac{-D\epsilon^2}{4d+8}\right)$, by [75]

$$\sup_{\mathbf{x}_i,\mathbf{x}_j\in\mathcal{X}}\left|\mathbf{z}_p^\top(\mathbf{x}_i)\mathbf{z}_p(\mathbf{x}_j) - \kappa_p(\mathbf{x}_i,\mathbf{x}_j)\right| < \epsilon \tag{3.58}$$

where $\epsilon > 0$ is a given constant, D the number of features, while d represents the dimension of $\mathbf{x}$, and $\sigma_p^2 := \mathbb{E}_p[\|\mathbf{v}\|^2]$ is the second-order moments of the RF vector norm. Henceforth, for the optimal function estimator (3.26) in $\mathcal{H}_p$ denoted by $f_p^*(\mathbf{x}) := \sum_{t=1}^{T}\alpha_{p,t}^*\kappa_p(\mathbf{x},\mathbf{x}_t)$, and its RF-based approximant $\check{f}_p^* := \sum_{t=1}^{T}\alpha_{p,t}^*\mathbf{z}_p^\top(\mathbf{x})\mathbf{z}_p(\mathbf{x}_t) \in \mathcal{F}_p$, we have

$$\begin{aligned}\left|\sum_{t=1}^{T}\mathcal{L}_t\left(\check{f}_p^*(\mathbf{x}_t)\right) - \sum_{t=1}^{T}\mathcal{L}_t\left(f_p^*(\mathbf{x}_t)\right)\right| &\overset{(a)}{\leq} \sum_{t=1}^{T}\left|\mathcal{L}_t\left(\check{f}_p^*(\mathbf{x}_t)\right) - \mathcal{L}_t(f_p^*(\mathbf{x}_t))\right|\\ &\overset{(b)}{\leq} \sum_{t=1}^{T} L\left|\sum_{t'=1}^{T}\alpha_{p,t'}^*\mathbf{z}_p^\top(\mathbf{x}_{t'})\mathbf{z}_p(\mathbf{x}_t) - \sum_{t'=1}^{T}\alpha_{p,t'}^*\kappa_p(\mathbf{x}_{t'},\mathbf{x}_t)\right|\end{aligned}$$

$$\overset{(c)}{\leq} \sum_{t=1}^{T} L \sum_{t'=1}^{T} |\alpha^*_{p,t'}| \left| \mathbf{z}_p^\top(\mathbf{x}_{t'})\mathbf{z}_p(\mathbf{x}_t) - \kappa_p(\mathbf{x}_{t'}, \mathbf{x}_t) \right| \tag{3.59}$$

where (a) follows from the triangle inequality; (b) uses the Lipschitz continuity of the loss, and (c) is due to the Cauchy-Schwarz inequality. Combining with (3.58), yields

$$\left| \sum_{t=1}^{T} \mathcal{L}_t(\check{f}_p^*(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(f_p^*(\mathbf{x}_t)) \right| \leq \sum_{t=1}^{T} L\epsilon \sum_{t'=1}^{T} |\alpha^*_{p,t'}| \leq \epsilon LTC, \text{ w.h.p.} \tag{3.60}$$

where the equality follows from $C := \max_p \sum_{t=1}^{T} |\alpha^*_{p,t}|$. Under the kernel bounds in (as3), the uniform convergence in (3.58) implies that $\sup_{\mathbf{x}_t, \mathbf{x}_{t'} \in \mathcal{X}} \mathbf{z}_p^\top(\mathbf{x}_t)\mathbf{z}_p(\mathbf{x}_{t'}) \leq 1 + \epsilon$, w.h.p., which in turn leads to

$$\left\| \boldsymbol{\theta}_p^* \right\|^2 := \left\| \sum_{t=1}^{T} \alpha^*_{p,t} \mathbf{z}_p(\mathbf{x}_t) \right\|^2 = \left| \sum_{t=1}^{T} \sum_{t'=1}^{T} \alpha^*_{p,t} \alpha^*_{p,t'} \mathbf{z}_p^\top(\mathbf{x}_t)\mathbf{z}_p(\mathbf{x}_{t'}) \right| \leq (1+\epsilon)C^2 \tag{3.61}$$

where we again used the definition of C.

Lemma 4 together with (3.60) and (3.61) leads to the regret of the proposed Raker algorithm relative to the best static function in $\mathcal{H}_p$, that is given by

$$\begin{aligned}
&\sum_{t=1}^{T} \mathcal{L}_t\bigg(\sum_{p=1}^{P} w_{p,t} \hat{f}_{p,t}(\mathbf{x}_t) \bigg) - \sum_{t=1}^{T} \mathcal{L}_t(f_p^*(\mathbf{x}_t)) \\
=& \sum_{t=1}^{T} \mathcal{L}_t\bigg(\sum_{p=1}^{P} w_{p,t} \hat{f}_{p,t}(\mathbf{x}_t) \bigg) - \sum_{t=1}^{T} \mathcal{L}_t\left(\check{f}_p^*(\mathbf{x}_t)\right) + \sum_{t=1}^{T} \mathcal{L}_t\left(\check{f}_p^*(\mathbf{x}_t)\right) - \sum_{t=1}^{T} \mathcal{L}_t(f_p^*(\mathbf{x}_t)) \\
\leq& \frac{\ln P}{\eta} + \frac{\eta L^2 T}{2} + \eta T + \frac{(1+\epsilon)C^2}{2\eta} + \epsilon LTC, \text{ w.h.p.}
\end{aligned} \tag{3.62}$$

which completes the proof of Theorem 1.

3.7.3 Proof of Lemma 5

Using Theorem 1 with $\eta = \epsilon = \mathcal{O}(1/\sqrt{T})$, it holds w.h.p. that

$$\sum_{t=1}^{T} \mathcal{L}_t\Big(\sum_{p=1}^{P} w_{p,t}\hat{f}_{p,t}(\mathbf{x}_t)\Big) - \sum_{t=1}^{T} \mathcal{L}_t(f_{p^*}^*(\mathbf{x}_t)) \leq \Big(\ln P + \frac{C^2}{2} + \frac{L^2}{2} + LC\Big)\sqrt{T} := c_0\sqrt{T} \tag{3.63}$$

where $p^* := \arg\min_{p\in\mathcal{P}} \sum_{t=1}^{T} \mathcal{L}_t\left(\hat{f}_p^*(\mathbf{x}_t)\right)$. At the end of interval I, we then deduce that the static regret of the Raker learner $\mathcal{A}_I$ is (cf. (3.36))

$$\text{Reg}^{\text{s}}_{\mathcal{A}_I}(|I|) = \sum_{t\in I} \mathcal{L}_t\left(\hat{f}_t^{(I)}(\mathbf{x}_t)\right) - \sum_{t\in I} \mathcal{L}_t(f_I^*(\mathbf{x}_t)) \leq c_0\sqrt{|I|}, \text{ w.h.p.} \tag{3.64}$$

where $\hat{f}_t^{(I)}(\mathbf{x}_t)$ is defined in (3.33), and $f_I^* \in \arg\min_{f\in\bigcup_{p\in\mathcal{P}}\mathcal{H}_p} \sum_{t\in I} \mathcal{L}_t(f(\mathbf{x}_t))$. To this end, we sketch the main steps leading to Lemma 5 as follows.

For every interval I, the static regret of the AdaRaker can be decomposed as

$$\begin{aligned}\text{Reg}^{\text{s}}_{\text{AdaRaker}}(|I|) &= \sum_{t\in I} \mathcal{L}_t\left(\hat{f}_t(\mathbf{x}_t)\right) - \sum_{t\in I} \mathcal{L}_t\left(\hat{f}_t^{(I)}(\mathbf{x}_t)\right) + \sum_{t\in I} \mathcal{L}_t\left(\hat{f}_t^{(I)}(\mathbf{x}_t)\right) - \sum_{t\in I} \mathcal{L}_t(f_I^*(\mathbf{x}_t)) \\ &:= \mathcal{R}_1 + \mathcal{R}_2\end{aligned} \tag{3.65}$$

where the first two sums in (3.65) represented by $\mathcal{R}_1$ capture the regret of the Ada-Raker learner $\mathcal{A}$ relative to the Raker learner $\mathcal{A}_I$; while the last two sums in (3.65) forming $\mathcal{R}_2$ denote the static regret of $\mathcal{A}_I$ on this interval. Notice that $\mathcal{R}_2$ directly follows from (3.64), while $\mathcal{R}_1$ can be bounded following the same steps in Lemma 7. Different from the kernel selections however, the crux is that the number of Raker learners (experts) $|\mathcal{I}(t)|$ is time-varying.

A tight bound can be derived via the *Sleeping Experts* reformulation of [93, 103], where an expert that has never appeared is thought of as being *asleep* for all previous rounds. For a looser bound, we assume the experts (instances $\{\mathcal{A}_I\}$) ever appeared until t are all active; that is, the total number of experts is upper bounded by $t\log t$, since at most $\log t$ experts are run during time t. Using (3.48)-(3.55), we have that

$$\mathcal{R}_1 \leq \eta^{(I)}|I| + \frac{\ln(t\log t)}{\eta^{(I)}} = \sqrt{|I|}\,(1 + \ln t + \ln(\log t)) \leq \sqrt{|I|}\,(1 + 2\ln t) \tag{3.66}$$

where $\eta^{(I)} := 1/\sqrt{|I|}$, and $\ln(\log t) \leq \ln(t)$. With (3.64), for any interval $I \in \mathcal{I}$, we have

$$\mathrm{Reg}^{\mathrm{s}}_{\mathrm{AdaRaker}}(|I|) = \sqrt{|I|}\,(1 + c_0 + 2\ln t) \leq \sqrt{|I|}\,(1 + c_0 + 2\ln T)\,. \tag{3.67}$$

Since the static regret bound (3.65) holds only at the end of such interval, the bound (3.67) only holds for those intervals (collected in $\mathcal{I}$) (re)initializing Raker instance $\mathcal{A}_I$.

The next step is to show that (3.67) holds for any interval $I \subseteq \mathcal{T}$, possibly $I \notin \mathcal{I}$. This is possible whenever the interval set $\mathcal{I}$ is properly designed, e.g., the interval partition given in Section 4.1. For any interval I, define the set of subintervals covered by I as $\mathcal{I}\|I := \{I' \subseteq I, I' \in \mathcal{I}\}$. As argued in [93, Lemma 5], interval I can be partitioned into two sequences of *non-overlapping* but *consecutive* intervals, given by $\{I_{-m}, \ldots, I_0\} \subseteq \mathcal{I}\|I$ and $\{I_1, \ldots, I_n\} \subseteq \mathcal{I}\|I$, the lengths of which satisfy $|I_{k+1}|/|I_k| \leq 1/2$, $\forall k \in [1, n-1]$ and $|I_k|/|I_{k+1}| \leq 1/2$, $\forall k \in [-m, -1]$. Therefore, we have (using $\sum_{k=1}^{\infty} \sqrt{2^{-k}T_0} \leq 4\sqrt{T_0}$)

$$\mathrm{Reg}^{\mathrm{s}}_{\mathrm{AdaRaker}}(|I|) = \sum_{k=1}^{n-1} \mathrm{Reg}^{\mathrm{s}}_{\mathrm{Ada}}(|I_k|) + \sum_{k=-m}^{-1} \mathrm{Reg}^{\mathrm{s}}_{\mathrm{Ada}}(|I_k|) \leq C_0\sqrt{|I|} + C_1 \ln T \sqrt{|I|} \tag{3.68}$$

where the inequality follows from (3.67) with $|I|$ replaced by $|I_k|$ ($\leq |I|$), and C_0, C_1 are constants depending on c_0. This completes the proof of Lemma 5.

3.7.4 Proof of Theorem 2

To start, the dynamic regret in (3.34) can be decomposed as

$$\mathrm{Reg}^{\mathrm{d}}_{\mathcal{A}}(T) := \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(f^*(\mathbf{x}_t)) + \sum_{t=1}^{T} \mathcal{L}_t(f^*(\mathbf{x}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(f^*_t(\mathbf{x}_t)) \tag{3.69}$$

where $f^*(\cdot)$ is the best fixed function in (3.26), and $f^*_t(\cdot)$ is the best dynamic function in (3.35), both of which belong to the union of spaces $\bigcup_{p\in\mathcal{P}} \mathcal{H}_p$. In (3.69), the first difference of sums is the static regret of AdaRaker, while the second difference of sums is the relative loss between the best fixed function and the best dynamic solution in the common space.

Intuitively, if the time horizon T is large, then the average static regret will become small, but the gap between the two benchmarks is large. With the insights gained from [71, 104], T

essentially trades off the values of two terms. Thus, splitting $\mathcal{T}$ into sub-horizons $\{\mathcal{T}_s\}, s = 1, \dots, \lfloor T/\Delta T \rfloor$, each having length ΔT, the dynamic regret of AdaRaker can be bounded by

$$\begin{aligned}\mathrm{Reg}^{\mathrm{d}}_{\mathrm{AdaRaker}}(T) &= \sum_{s=1}^{\lfloor T/\Delta T \rfloor} \sum_{t \in \mathcal{T}_s} \Big(\mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \mathcal{L}_t(f^*_{\mathcal{T}_s}(\mathbf{x}_t))\Big) + \sum_{s=1}^{\lfloor T/\Delta T \rfloor} \sum_{t \in \mathcal{T}_s} \big(\mathcal{L}_t(f^*_{\mathcal{T}_s}(\mathbf{x}_t)) - \mathcal{L}_t(f^*_t(\mathbf{x}_t))\big) \\ &:= \sum_{s=1}^{\lfloor T/\Delta T \rfloor} \mathcal{R}_1 + \sum_{s=1}^{\lfloor T/\Delta T \rfloor} \mathcal{R}_2 \end{aligned} \tag{3.70}$$

where the first sum over $\mathcal{T}_s$ we define as $\mathcal{R}_1$ can be bounded under AdaRaker from Lemma 5, while the second sum over $\mathcal{T}_s$ that we define as $\mathcal{R}_2$ depends on the variability of the environments $\mathcal{V}(\{\mathcal{L}_t\})$, can be bounded by [71, Prop. 2]

$$\mathcal{R}_2 \leq 2\Delta T \mathcal{V}(\{\mathcal{L}_t\}_{t \in \mathcal{T}_s}). \tag{3.71}$$

Together with Lemma 5, it follows that

$$\begin{aligned}\mathrm{Reg}^{\mathrm{d}}_{\mathrm{AdaRaker}}(T) &\leq \sum_{s=1}^{\lfloor T/\Delta T \rfloor} \Big((C_0 + C_1 \ln T)\sqrt{\Delta T} + 2\Delta T \mathcal{V}(\{\mathcal{L}_t\}_{t \in \mathcal{T}_s})\Big) \\ &= (C_0 + C_1 \ln T)\frac{T}{\sqrt{|\Delta T|}} + 2|\Delta T|\mathcal{V}(\{\mathcal{L}_t\}_{t=1}^T), \text{ w.h.p.} \end{aligned} \tag{3.72}$$

Since (3.37) in Lemma 5 holds for any interval $\Delta T \subseteq \mathcal{T}$, after selecting ΔT so that $|\Delta T| = \big(T/\mathcal{V}(\{\mathcal{L}_t\}_{t=1}^T)\big)^{\frac{2}{3}}$, we arrive at

$$\mathrm{Reg}^{\mathrm{d}}_{\mathrm{AdaRaker}}(T) \leq (C_0 + C_1 \ln T) T^{\frac{2}{3}} \mathcal{V}^{\frac{1}{3}}(\{\mathcal{L}_t\}_{t=1}^T) + 2T^{\frac{2}{3}} \mathcal{V}^{\frac{1}{3}}(\{\mathcal{L}_t\}_{t=1}^T), \text{ w.h.p.} \tag{3.73}$$

which completes the proof of Theorem 2.

3.7.5 Proof of Theorem 3

Suppose that the m-switching dynamic solution $\{\check{f}^*_t\}$ changes at slots $t_1 = 1, \dots, t_m$, and define the m sub-intervals that partition $\mathcal{T} := \{1, \dots, T\}$ as $\mathcal{T}_1 := [1, t_2 - 1]$, $\mathcal{T}_2 := [t_2, t_3 - 1], \dots$, and $\mathcal{T}_m := [t_m, T]$. To use the bound in Lemma 5, we decompose the regret of AdaRaker relative to the m-switching dynamic solution $\{\check{f}_t^*\}$ by

$$\begin{aligned}\mathrm{Reg}^m_{\mathrm{AdaRaker}}(T) &\overset{(a)}{=} \sum_{s=1}^{m}\sum_{t\in\mathcal{T}_s}\Big(\mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \mathcal{L}_t(\check{f}^*_{t_s}(\mathbf{x}_t))\Big)\\ &\overset{(b)}{\leq} \sum_{s=1}^{m}\sum_{t\in\mathcal{T}_s}\Big(\mathcal{L}_t(\hat{f}_t(\mathbf{x}_t)) - \mathcal{L}_t(f^*_{\mathcal{T}_s}(\mathbf{x}_t))\Big) \qquad (3.74)\end{aligned}$$

where (a) holds because the definition of $\{\check{f}_t^*\}$ in (3.40) implies that $\check{f}_t^* = \check{f}^*_{t_s}$, $\forall t \in \mathcal{T}_s$, and (b) because the best fixed function is given by $f^*_{\mathcal{T}_s} \in \arg\min_{f\in f\in\bigcup_{p\in\mathcal{P}}\mathcal{H}_p}\sum_{t\in\mathcal{T}_s}\mathcal{L}_t(f(\mathbf{x}_t))$. Therefore, using the regret bound of Lemma 5 in (3.37), we have

$$\mathrm{Reg}^m_{\mathrm{AdaRaker}}(T) \leq \sum_{s=1}^{m}\mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}}(|\mathcal{T}_s|) \leq (C_0 + C_1\ln T)\sum_{s=1}^{m}\sqrt{|\mathcal{T}_s|}. \qquad (3.75)$$

Holder's inequality further implies that

$$\begin{aligned}\mathrm{Reg}^m_{\mathrm{AdaRaker}}(T) &\leq (C_0 + C_1\ln T)\left(\sum_{s=1}^{m}(1)^2\right)^{\frac{1}{2}}\left(\sum_{s=1}^{m}\left(\sqrt{|\mathcal{T}_s|}\right)^2\right)^{\frac{1}{2}}\\ &\leq (C_0 + C_1\ln T)\sqrt{Tm},\ \text{w.h.p.} \qquad (3.76)\end{aligned}$$

which completes the proof of Theorem 3.

Chapter 4

Tensor-based network topology identification

4.1 Introduction

The study of networks and network phenomena has recently emerged as a major catalyst for collectively understanding the behavior of complex systems [9, 105, 106]. Such systems are ubiquitous, and commonly arise in both natural and man-made settings. For example, online interactions over the web are commonly facilitated through social networks such as Facebook and Twitter, while sophisticated brain functions are the result of vast interactions within complex neuronal networks; see e.g., [107] and references therein. Other networks naturally emerge in settings as diverse as financial markets, genomics and proteomics, power grids, and transportation systems, to name just a few.

While some of these networks are directly observable, due to e.g., presence of physical or engineered links between nodes, most complex networks have hidden topologies, which must first be inferred in order to conduct meaningful network analytics [9, Ch. 7]; see also [108–110]. Prominent among these are SEMs, a family of statistical approaches for causal (a.k.a., path) analysis in complex systems, with several applications specifically tailored to graph topology inference; see e.g., [28, 111–113]. In a nutshell, SEMs capture the relationship between observed nodal processes or measurements, and the unknown causal network. The key contribution of SEMs is two-fold: a) they are conceptually simple, often resorting to tractable linear models;

and b) SEMs explicitly account for the role played by exogenous or confounding inputs in observed nodal processes, which turn out to be critical in resolving directional ambiguities [114].

In settings where measurement of exogenous inputs is costly or impractical, contemporary SEMs are quite limited with regard to unique identification of hidden network topologies. For example, in financial networks comprising stocks as nodes and their interdependencies as links, publicly-traded stock prices (endogenous) are known to depend on stock purchases (exogenous) by investors, whose details are often unknown to the public for privacy reasons. On the other hand, each publicly-traded company may broadcast monthly statistical summaries of purchases of its stock. Assuming that such statistical information is known or obtainable, the present chapter advocates novel approaches that capitalize on factorization of carefully constructed *tensors*, or multi-modal arrays. As demonstrated later, inference of the network topology is shown possible under reasonable conditions, using only correlation information of the exogenous inputs. The crux of our novel framework lies in positing that exogenous inputs exhibit piecewise-stationary correlations, from which three-way tensors are constructed using a special instance of SEMs.

By leveraging the well-known parallel factor (PARAFAC) tensor decomposition [115], it is shown that edge connectivity information is captured through one of the factors, while identifiability of the network topology is guaranteed due to uniqueness of the factorization. Interestingly, casting the problem as tensor decomposition also opens up opportunities to *blindly* estimate both the unknown topology and *local* correlation matrices of the exogenous inputs; see also [116, 117]. PARAFAC decomposition is a powerful tool for multilinear algebra introduced by [118], and its merits have been permeated within diverse application domains [119], e.g., wireless communications [120], blind source separation [121, 122], as well as community detection on graphs [123, 124]. The present chapter broadens these well-documented merits to tasks involving network topology inference. Numerical tests on simulated and real data corroborate the efficacy of the novel approach.

Since most real-world networks are time-varying, the advocated tensor-based approach is accordingly extended to track topology changes. Moreover, nodal data are often acquired in real-time streams, rendering batch inference algorithms impractical. Toward satisfying the dual need to mitigate batch computational overhead, and track dynamic topologies, an online variant of the novel algorithm is developed. Motivated by the adaptive PARAFAC decomposition [44, 125], a novel real-time estimator is put forth to track the topology-revealing tensor factors, using second-order statistics of the exogenous inputs.

To place this work in context, several prior studies have focused on tracking time-varying networks from nodal processes. For example, dynamic information diffusion networks were tracked via maximum likelihood estimators in [126], while a sparse piecewise stationary graphical model was put forth to track undirected networks in [127]. Dynamic SEMs were also advocated for inference of dynamic and directed cascade networks in [112]. More recent work in [128] resorted to hidden Markov models (HMMs) to track diffusion links.

PARAFAC decompositions have previously been advocated in e.g., blind source separation (BSS) tasks, which separate source signals from their mixed observations; see e.g., [121, 122] [129]. However, tensor-based SEMs present unique challenges not encountered in traditional BSS problems addressed by these prior works, namely: i) network topologies are not directly revealed by tensor decomposition factors, which suggests leveraging properties inherent to SEMs; and ii) the inherent scaling and permutation ambiguities are affordable compromises in BSS, but intolerable in the context of topology identification. Identifiability conditions developed in this chapter aim to address these challenges. Tensor factorizations have also recently been adopted in network analytics, graph mining, and sensor networks. For instance, several community detection approaches leverage the flexibility of tensors to capture more complex connectivity patterns such as *cliques* and *egonets*; see e.g., [123, 130], and [131]. On the other hand, [132] puts forth a tensor-based blind identification strategy to jointly recover transmitted data, and the network connectivity in collaborative wireless sensor networks. Nevertheless, the approach advocated by [132] is markedly different from the present one because; i) it relies on coding sensor data with a *known* coding matrix; and ii) it can only guarantee identifiability of unweighted graphs.

The rest of this section is organized as follows. Preliminaries and a formal statement of the problem are given in Section 4.2, while Section 4.3 casts the problem as a tensor factorization. Section 4.4 presents identifiability results for the proposed framework, while a topology tracking algorithm is developed in Section 4.5. Finally, results of corroborating numerical tests on both synthetic and real data are presented in Section 4.6, while concluding remarks and a discussion of ongoing and future directions are given in Section 4.7.

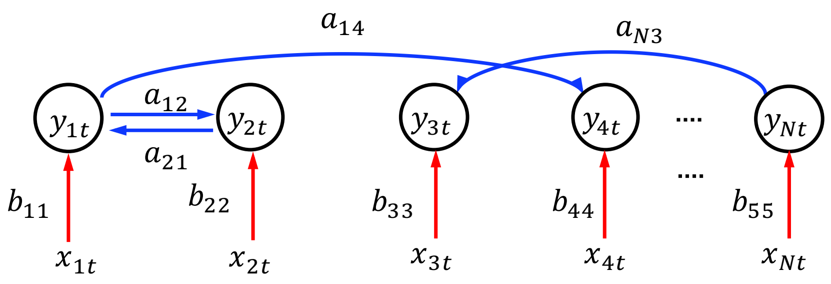


Figure 4.1: An N-node directed network (blue links), with the t-th samples of endogenous measurements per node. SEMs explicitly account for exogenous inputs (red arrows), upon which endogenous variables may depend, in addition to the underlying topology.

4.2 Preliminaries and Problem Statement

Consider a network $\mathcal{G}(\mathcal{V}, \mathcal{E})$ that comprises N nodes, with its topology captured by an unknown adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$. Let a_{ij} denote entry (i, j) of $\mathbf{A}$, which is nonzero only if there is an edge between nodes i and j; see Figure 4.1. It will generally be assumed that $\mathcal{G}$ is a directed graph, that is $\mathbf{A}$ is a non-symmetric matrix $(\mathbf{A} \neq \mathbf{A}^\top)$.

Suppose the network abstracts a complex system with measurable inputs and an observable output process that propagates over the network following directed links. Let x_{it} denote the input to node i at slot t, and y_{it} the t-th observation of the propagating process measured at node i. In the context of brain networks, y_{it} could represent the t-th time sample of an electroencephalogram (EEG), or functional magnetic resonance imaging (fMRI) measurement at region i, while x_{it} could be a controlled stimulus that affects a specific region of the brain. In social networks (e.g., *Twitter* or *Facebook*) over which information diffuses, y_{it} could represent the timestamp when subscriber i tweeted or shared a specific viral story, while x_{it} could measure their level of interest in the story; see also [112].

In general, SEMs postulate that y_{it} depends on two classes of variables, namely: i) measurements of the diffusing process $\{y_{jt}\}_{j \neq i}$ (a.k.a. *endogenous* variables); and ii) external inputs x_{it} (a.k.a. *exogenous* variables). Most contemporary SEM approaches posit that y_{it} depends linearly on both $\{y_{jt}\}_{j \neq i}$ and x_{it}, during interval t; that is [112, 133]

$$y_{it} = \underbrace{\sum_{j \neq i} a_{ij} y_{jt}}_{\text{endogenous term}} + \underbrace{b_{ii} x_{it}}_{\text{exogenous term}} + e_{it} \tag{4.1}$$

where $[\mathbf{A}]_{ij} := a_{ij}$, and e_{it} denotes an "error" term that captures unmodeled dynamics. Although (4.1) apparently captures "instantaneous" (within each slot) interactions, depending on the application, the per-slot duration can be chosen so that e.g., a measurement at the beginning of slot j can causally affect the measurement at node i at the end of the same slot when the network has reached a steady state. The coefficients $\{a_{ij}\}$ and $\{b_{ii}\}$ are unknown, and $a_{ij} \neq 0$ signifies that a directed edge from j to i is present. Collecting nodal measurements $\mathbf{y}_t{:=}[y_{1t} \ldots y_{Nt}]^\top$, and $\mathbf{x}_t{:=}[x_{1t} \ldots x_{Nt}]^\top$ per slot t, and temporarily assuming that $e_{jt} = 0$, the noise-free version of (4.1) can be compactly written as

$$\mathbf{y}_t = \mathbf{A}\mathbf{y}_t + \mathbf{B}\mathbf{x}_t \tag{4.2}$$

where $[\mathbf{A}]_{ii} = 0$ and $\mathbf{B} := \text{Diag}(b_{11}, \ldots, b_{NN})$ denotes a diagonal coefficient matrix.

Note that with $\mathbf{B}$ diagonal, (4.1) implicitly assumes that each node is associated with a single exogenous input. In fact, it is possible to generalize (4.1) to settings where a single exogenous input may be applied to several nodes, or where a single node may be the recipient of multiple inputs. This amounts to relaxing the restriction on $\mathbf{B}$, allowing it to take values from the set of non-diagonal square matrices. In addition, in more general SEMs $\mathbf{x}_t$ and $\mathbf{y}_t$ are indirectly observed latent variables, each adhering to *measurement* models, namely $\mathbf{u}_{yt} = \mathbf{C}_y\mathbf{y}_t + \boldsymbol{\delta}_{yt}$ and $\mathbf{u}_{xt} = \mathbf{C}_x\mathbf{x}_t + \boldsymbol{\delta}_{xt}$, with corresponding noise terms $\boldsymbol{\delta}_{yt}$ and $\boldsymbol{\delta}_{xt}$; see e.g., [133] for details. In this case, the noisy version ($\mathbf{y}_t = \mathbf{A}\mathbf{y}_t + \mathbf{B}\mathbf{x}_t + \mathbf{e}_t$) of (4.2) is often referred to as the *structural model*. This chapter deals with settings where $\mathbf{x}_t$ and $\mathbf{y}_t$ are directly observable, and there is no extra measurement model. Different from conventional SEM settings where exogenous inputs $\{\mathbf{x}_t\}_{t=1}^T$ are assumed known explicitly, the present chapter only assumes partial knowledge of second-order statistics $\{\mathbf{x}_t\}$. The problem statement can now be formally stated as follows.

Problem statement: Given second-order statistics of $\{\mathbf{y}_t\}_{t=1}^T$, and either full or partial second-order statistics of $\{\mathbf{x}_t\}_{t=1}^T$, the goal is to recover and track the underlying directed network topology $\mathbf{A}$.

4.3 A Tensor Factorization Approach

Building upon (4.1), this section puts forth a novel tensor factorization approach to unveil the hidden network topology. To this end, the following assumptions are adopted.

(as0) Exogenous data $\{\mathbf{x}_t^{(m)}\}$ are *piecewise-stationary* over time segments $t \in [\tau_m, \tau_{m+1} - 1], m = 1, \ldots, M+1$, each with a fixed correlation matrix $\mathbf{R}_m^x := \mathbb{E}\{\mathbf{x}_t^{(m)}(\mathbf{x}_t^{(m)})^\top\}$;
(as1) Entries of $\mathbf{x}_t$ are zero mean and uncorrelated per t; that is, $\mathbb{E}\{x_{it}x_{jt}\} = 0, \forall i \neq j$;
(as2) Matrix $(\mathbf{I} - \mathbf{A})$ is invertible; and
(as3) Matrix $\mathbf{B}$ is diagonal with nonzero diagonal entries.
Under (as0) and (as2), it is possible to rewrite (4.2) as

$$\mathbf{y}_t = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}\mathbf{x}_t = \boldsymbol{\mathcal{A}}\mathbf{x}_t \tag{4.3}$$

where $\boldsymbol{\mathcal{A}} := (\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$, and superscript (m) has been dropped with the understanding that t stays within one segment, and thus (4.3) holds $\forall m$. The *per segment* correlation matrix $\mathbf{R}_m^y := \mathbb{E}\{\mathbf{y}_t\mathbf{y}_t^\top\}$ is thus given by (cf. (4.3))

$$\mathbf{R}_m^y = \boldsymbol{\mathcal{A}}\mathbf{R}_m^x\boldsymbol{\mathcal{A}}^\top, \ \ t \in [\tau_m, \tau_{m+1} - 1]. \tag{4.4}$$

Under (as1), one can express (4.4) as the weighted sum of rank-one matrices as

$$\mathbf{R}_m^y = \boldsymbol{\mathcal{A}}\text{Diag}(\boldsymbol{\rho}_m^x)\boldsymbol{\mathcal{A}}^\top = \sum_{i=1}^{N} \rho_{mi}^x \boldsymbol{\alpha}_i \boldsymbol{\alpha}_i^\top \tag{4.5}$$

where $\boldsymbol{\alpha}_i$ denotes the ith column of $\boldsymbol{\mathcal{A}}$, and $\boldsymbol{\rho}_m^x := [\rho_{m1}^x \ldots \rho_{mN}^x]^\top$, with $\rho_{mi}^x := \mathbb{E}(x_{it}^2)$, for $t \in [\tau_m, \tau_{m+1} - 1]$.

Consider the three-way tensor $\underline{\mathbf{R}}^y \in \mathbb{R}^{N \times N \times M}$, constructed by setting the m-th slice $[\underline{\mathbf{R}}^y]_{:,:,m} = \mathbf{R}_m^y$. Letting $\alpha_{ji}\beta_{ki}\gamma_{li}$ denote the (j,k,l) entry of the tensor outer product $\boldsymbol{\alpha}_i \circ \boldsymbol{\beta}_i \circ \boldsymbol{\gamma}_i$, where $\alpha_{ji} := [\boldsymbol{\alpha}_i]_j$ (resp. β_{ik} and γ_{il}), it turns out that $\underline{\mathbf{R}}^y$ can be written as (see also Figure 4.2)

$$\underline{\mathbf{R}}^y = \sum_{i=1}^{N} \boldsymbol{\alpha}_i \circ \boldsymbol{\alpha}_i \circ \mathbf{r}_i^x \tag{4.6}$$

with entry (j,k,l) given by

$$[\underline{\mathbf{R}}^y]_{jkl} = \sum_{i=1}^{N} \alpha_{ji}\alpha_{ki}r_{li}^x \tag{4.7}$$

where $\mathbf{r}_i^x := [\rho_{1i}^x \dots \rho_{Mi}^x]^\top$. Interestingly, (4.6) amounts to the so-termed partial symmetric PARAFAC decomposition of $\underline{\mathbf{R}}^y$ into factor matrices $\boldsymbol{\mathcal{A}}$, $\boldsymbol{\mathcal{A}}$, and $\mathbf{R}^x := [\mathbf{r}_1^x \dots \mathbf{r}_N^x] \in \mathbb{R}^{M\times N}$; see e.g., [115]. Although $\mathbf{R}_m^y$ is generally unknown, it can be readily estimated using sample averaging as

$$\hat{\mathbf{R}}_m^y = \frac{1}{\tau_{m+1} - \tau_m} \sum_{t=\tau_m}^{\tau_{m+1}-1} \mathbf{y}_t \mathbf{y}_t^\top, \quad m = 1, \dots, M \tag{4.8}$$

from endogenous measurements.

The present chapter relies on this three-way tensor constructed from second-order statistics of the nodal measurements, and leverages the uniqueness properties inherent to PARAFAC decompositions to identify the hidden network topology; see e.g., [134] for key uniqueness results. Indeed, a number of standard PARAFAC decomposition algorithms can be adopted to estimate $\boldsymbol{\mathcal{A}}$; e.g., via alternating least-squares (ALS) iterations. Under reasonable conditions, it will be possible to recover $\mathbf{A}$, once $\boldsymbol{\mathcal{A}}$ has been found. The next proposition formally states the sufficient conditions required to uniquely identify $\mathbf{A}$, after determing of $\boldsymbol{\mathcal{A}}$ from the PARAFAC decomposition.

Proposition 3: *If (as2) and (as3) hold, then* $\mathbf{A}$ *can be uniquely expressed in terms of* $\boldsymbol{\mathcal{A}}$ *as* $\mathbf{A} = \mathbf{I} - \left(\mathit{Diag}(\boldsymbol{\mathcal{A}}^{-1})\right)^{-1} \boldsymbol{\mathcal{A}}^{-1}$.

Proof: See Appendix 4.8.1.

Regarding the decomposition in (4.6), one can make the following important observations: (i) rank($\underline{\mathbf{R}}^y$) $= N$, where tensor rank is defined as the number of summands in (4.6); (ii) two factors of $\underline{\mathbf{R}}^y$ are identical; and (iii) the tensor formulation in (4.6) only involves the second-order statistics $\{\mathbf{r}_i^x\}$, instead of explicit knowledge of the exogenous inputs $\{\mathbf{x}_t\}_{t=1}^T$.

Consider $\mathbf{R}_\Omega^x$ known a priori, where Ω denotes the index set of the available entries of $\mathbf{R}^x$, i.e., $[\mathbf{R}_\Omega^x]_{i,j} = r_{ij}^x$ for $(i,j) \in \Omega$. Given noisy tensor data, these considerations (i)–(iii) prompt the next criterion for obtaining the wanted factors

$$(\hat{\mathbf{Z}}_1, \hat{\mathbf{Z}}_2, \hat{\mathbf{Z}}_3) = \arg \min_{\mathbf{Z}_1, \mathbf{Z}_2, \mathbf{Z}_3} \left\| \underline{\mathbf{R}}^y - \sum_{n=1}^N \mathbf{z}_{1n} \circ \mathbf{z}_{2n} \circ \mathbf{z}_{3n} \right\|_F^2$$
$$\text{s.t.} \quad \mathbf{Z}_1 = \mathbf{Z}_2, \; [\mathbf{Z}_3]_{i,j} = [\mathbf{R}_\Omega^x]_{i,j}, \; \forall (i,j) \in \Omega \qquad \text{(P1)}$$

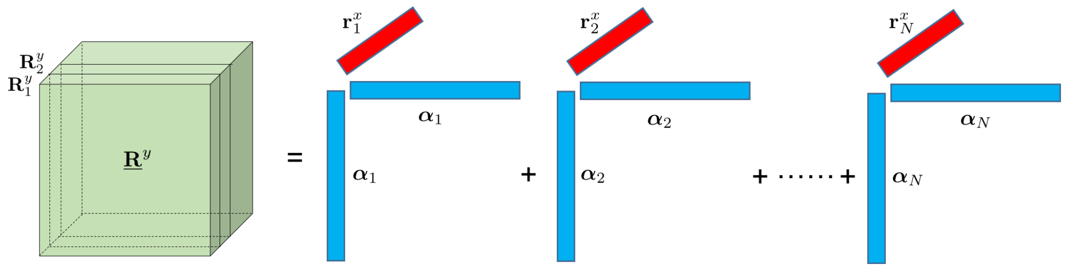


Figure 4.2: The tensor constructed by stacking the correlation matrices admits a PARAFAC decomposition comprising rank-one tensor outer products.

where $\mathbf{z}_{in}$ denotes the n-th column of matrix $\mathbf{Z}_i$. Note that (P1) can be solved via partially symmetric PARAFAC decomposition, even when noise is present, using e.g., the individual differences in multidimensional scaling [135]. Upon obtaining the estimated factors $\hat{\mathbf{Z}}_1, \hat{\mathbf{Z}}_2$ and $\hat{\mathbf{Z}}_3$, matrix $\hat{\mathbf{A}}$ can be found as (cf. Proposition 3)

$$\widehat{\boldsymbol{\mathcal{A}}} = \hat{\mathbf{Z}}_1 \tag{4.9}$$

$$\hat{\mathbf{A}} = \mathbf{I} - \left(\text{Diag}(\widehat{\boldsymbol{\mathcal{A}}}^{-1})\right)^{-1} \widehat{\boldsymbol{\mathcal{A}}}^{-1}. \tag{4.10}$$

Unlike [114] where explicit knowledge of the exogenous inputs is assumed to ensure model identifiability, our novel approach here establishes that knowledge of the second-order statistics captured through $\mathbf{R}^x$ could suffice. Detailed conditions under which the novel approach uniquely identifies the topology will be provided in Section 4.4. Algorithm 4b summarizes the resulting network topology inference scheme. It is assumed that one is given endogenous measurements $\{\mathbf{y}_t\}_{t=1}^T$, as well as $\mathbf{R}_\Omega^x$. It is also worth pointing out that S1 constructs $\underline{\mathbf{R}}^y$ from endogenous data using the sample correlation matrices in (4.8), since local correlation matrices $\{\mathbf{R}_m^y\}_{m=1}^M$ are not explicitly known. The prescribed threshold η in S4 is employed to determine the presence of edges. Its selection will be discussed in Section 4.6.

Remark 1: The PARAFAC decomposition generally assumes no prior knowledge about $\mathbf{R}^x$; that is, $\Omega = \varnothing$ in (P1). In principle, one can estimate the topology even without correlation information of the exogenous inputs. Interestingly, this amounts to blindly estimating the topology and exogenous correlation matrices, which is of considerable merit when measurement of external inputs is impossible, or rather costly.

Algorithm 4b Topology inference via tensor decomposition

```
Input: $\mathbf{R}^x_\Omega$, $\{\mathbf{y}_t\}$, $M$, $\eta$
S1. Tensor construction:
        Set $m$-th frontal slice of $\underline{\mathbf{R}}^y \in \mathbb{R}^{N\times N\times M}$ to
        $\widehat{\mathbf{R}}^y_m = \frac{1}{\tau_{m+1}-\tau_m}\sum_{t=\tau_m}^{\tau_{m+1}-1}\mathbf{y}_t\mathbf{y}_t^\top,\ m = 1,\ldots,M$
S2. PARAFAC decomposition via e.g., [135]:
        Solve (P1) to find $(\hat{\mathbf{Z}}_1, \hat{\mathbf{Z}}_2, \hat{\mathbf{Z}}_3)$
S3. SEM estimates for topology inference:
        $\widehat{\boldsymbol{\mathcal{A}}} = \hat{\mathbf{Z}}_1$
        $\hat{\mathbf{A}} = \mathbf{I} - \left(\text{Diag}(\widehat{\boldsymbol{\mathcal{A}}}^{-1})\right)^{-1}\widehat{\boldsymbol{\mathcal{A}}}^{-1}$
S4. Edge identification:
        $[\hat{\mathbf{A}}]_{ij} \neq 0$ if $[\hat{\mathbf{A}}]_{ij} > \eta$, otherwise $[\hat{\mathbf{A}}]_{ij} = 0, \forall(i,j)$
```

4.4 Identifiability issues

Although casting network topology identification task as a tensor decomposition problem leads to enhanced flexibility, one has to contend with identifiability issues common to both matrix and tensor factorizations. In order to establish identifiability conditions for $\mathbf{A}$ and $\mathbf{B}$, this section will first explore conditions under which $\boldsymbol{\mathcal{A}}$ is uniquely identifiable. To this end, a couple of definitions are in order.

Definition 1. The *Kruskal rank* of a matrix $\mathbf{Z} \in \mathbb{R}^{N\times M}$ (denoted hereafter as kr($\mathbf{Z}$)) is defined as the maximum number k such that *any* combination of k columns of $\mathbf{Z}$ constitutes a full rank submatrix.

Definition 2. *Essential* uniqueness of a tensor factorization refers to uniqueness up to scaling and permutation ambiguity.

With Definitions 1 and 2 in mind, consider PARAFAC decomposition for a three way tensor $\underline{\mathbf{P}} = (\mathbf{U}, \mathbf{V}, \mathbf{W})$. Theorem 4 establishes sufficient conditions for essential uniqueness of the tensor decomposition; see [136] and [134] for further details and a proof of the theorem.

Theorem 4 *Let* $(\mathbf{U}, \mathbf{V}, \mathbf{W})$ *denote the PARAFAC factors obtained by decomposing a three-way tensor* $\underline{\mathbf{P}}$ *into* K *rank-one tensors. If Kruskal's condition holds, namely,*

$$\text{kr}(\mathbf{U}) + \text{kr}(\mathbf{V}) + \text{kr}(\mathbf{W}) \geq 2K + 2 \tag{4.11}$$

and there exists an alternative set of matrices $(\bar{\mathbf{U}}, \bar{\mathbf{V}}, \bar{\mathbf{W}})$ *constituting a PARAFAC decomposition of* $\underline{\mathbf{P}}$*, then there exists a permutation matrix* $\mathbf{\Pi}$*, and diagonal scaling matrices* $\mathbf{\Lambda}_1$*,* $\mathbf{\Lambda}_2$*,* $\mathbf{\Lambda}_3$*, such that* $\mathbf{\Lambda}_1\mathbf{\Lambda}_2\mathbf{\Lambda}_3 = \mathbf{I}$*,* $\bar{\mathbf{U}} = \mathbf{U}\mathbf{\Pi}\mathbf{\Lambda}_1$ *,* $\bar{\mathbf{V}} = \mathbf{V}\mathbf{\Pi}\mathbf{\Lambda}_2$*, and* $\bar{\mathbf{W}} = \mathbf{W}\mathbf{\Pi}\mathbf{\Lambda}_3$*.*

Proof: See [136] for a general proof with complex entries.

As a prerequisite to identification of $\mathbf{A}$, the following proposition establishes essential uniqueness of $\boldsymbol{\mathcal{A}}$, based on the tensor-based interpretation advocated in the prequel.

Proposition 4: *If* $\mathrm{kr}(\mathbf{R}^x) \geq 2$*, then* $\boldsymbol{\mathcal{A}} := (\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$ *is uniquely identifiable up to a scaling and permutation ambiguity via PARAFAC decomposition of* $\underline{\mathbf{R}}^y$*.*

Proof: Upon recognizing that rank$(\underline{\mathbf{R}}^y) = N$ from (4.6), in order for (4.11) to hold, we need

$$2\mathrm{kr}(\boldsymbol{\mathcal{A}}) + \mathrm{kr}(\mathbf{R}^x) \geq 2N + 2. \tag{4.12}$$

Under (as2) and (as3), matrices $(\mathbf{I} - \mathbf{A})$ and $\mathbf{B}$ are invertible, which implies that $\boldsymbol{\mathcal{A}} = (\mathbf{I} - \mathbf{A})^{-1}\mathbf{B}$ is invertible, and hence $\mathrm{kr}(\boldsymbol{\mathcal{A}}) = N$. From (4.12), essential uniqueness can thus be guaranteed as long as $\mathrm{kr}(\mathbf{R}^x) \geq 2$, which completes the proof.

Note that essential uniqueness is not sufficient for identification of the hidden network topology, due to the inherent permutation and scaling ambiguities. To this end, we will subsequently pursue identifiability conditions for settings where $\mathbf{R}^x$ may be fully, or partially available, or even completely unavailable on a case-by-case basis.

4.4.1 Identifiability with fully known $\mathbf{R}^x$

First, we will explore identifiability of the topology when $\mathbf{R}^x$ is completely known, while highlighting the importance of information about exogenous inputs $\{\mathbf{x}_t\}$. It is worth mentioning that identifiability result for the general tensor decomposition with a known factor have been derived in [137].

Theorem 5 *If* $\mathbf{x}_t$ *and* $\mathbf{y}_t$ *obey the SEM in* (4.2)*, for all* $t = 1, \ldots$*, with* $\mathbf{A}$ *and* $\mathbf{B}$ *satisfying (as2) and (as3), respectively, and if* $\mathbf{R}^x$ *is known and satisfies* $\mathrm{kr}(\mathbf{R}^x) \geq 2$*, then* $\mathbf{A}$ *can be uniquely identified via Algorithm 4b.*

Proof: Suppose there is an alternative triplet $(\boldsymbol{\mathcal{A}}', \boldsymbol{\mathcal{A}}', \mathbf{R}^{x\prime})$, also decomposing $\underline{\mathbf{R}}^y$ into N rank-one tensors in (P1). Theorem 4 asserts that there is a permutation matrix $\mathbf{\Pi}$, and diagonal scaling matrices $\{\mathbf{\Lambda}_1, \mathbf{\Lambda}_2, \mathbf{\Lambda}_3\}$ so that

$$\mathbf{\Lambda}_1\mathbf{\Lambda}_2\mathbf{\Lambda}_3 = \mathbf{I} \tag{4.13}$$

and

$$\begin{aligned}
\boldsymbol{\mathcal{A}}' &= \boldsymbol{\mathcal{A}}\mathbf{\Pi}\mathbf{\Lambda}_1 && \text{(4.14a)}\\
\boldsymbol{\mathcal{A}}' &= \boldsymbol{\mathcal{A}}\mathbf{\Pi}\mathbf{\Lambda}_2 && \text{(4.14b)}\\
\mathbf{R}^{x\prime} &= \mathbf{R}^{x}\mathbf{\Pi}\mathbf{\Lambda}_3 && \text{(4.14c)}
\end{aligned}$$

where one can readily deduce from (4.14a) and (4.14b) that $\mathbf{\Lambda}_1 = \mathbf{\Lambda}_2$. On the other hand, when $\mathbf{R}^x$ is known a priori, i.e., $\mathbf{R}^x_\Omega = \mathbf{R}^x$, the constraint in (P1) yields $\mathbf{R}^{x\prime} = \mathbf{R}^x$. Consequently, (4.14c) can be written as

$$\mathbf{R}^x = \mathbf{R}^x\mathbf{\Pi}\mathbf{\Lambda}_3 \tag{4.15}$$

for which the following holds.

Lemma 8: *For permutation matrix $\mathbf{\Pi}$, scaling matrix $\mathbf{\Lambda}_3$, and $\mathbf{R}^x$ satisfying the inequality* $\mathrm{kr}(\mathbf{R}^x) \geq 2$, (4.15) *holds true if and only if*

$$\begin{aligned}
\mathbf{\Lambda}_3 &= \mathbf{I} && \text{(4.16a)}\\
\mathbf{\Pi} &= \mathbf{I}. && \text{(4.16b)}
\end{aligned}$$

Proof: See Appendix 4.8.2.

Next, substituting (4.16b) into (4.14a), and letting $\mathbf{\Lambda} = \mathbf{\Lambda}_1 = \mathbf{\Lambda}_2$, one obtains

$$\boldsymbol{\mathcal{A}}' = \boldsymbol{\mathcal{A}}\mathbf{\Lambda}. \tag{4.17}$$

for which the next lemma holds true.

Lemma 9: *If the PARAFAC solution obtained in S3 of Algorithm 4b satisfies $\widehat{\boldsymbol{\mathcal{A}}} = \boldsymbol{\mathcal{A}}\mathbf{\Lambda}$, then $\mathbf{A}$ can be uniquely identified; that is, $\hat{\mathbf{A}} = \mathbf{A}$.*

Proof: See Appendix 4.8.3.

Combining Lemma 9 with (4.17) completes the proof of Theorem 5.

Remark 2: The result of Lemma 8 can be obtained as a special case of that in [137], where identifiability results for general tensor decompositions with a known factor have been reported. Notwithstanding, identifiability of tensor factors is not identical to identifiability of network topology. As Lemma 9 asserts, in order to guarantee recovery of a unique topology, the tensor factor $\boldsymbol{\mathcal{A}}$ need to be identified just up to scaling ambiguity, which is different from "essential uniqueness" that can also afford ambiguity within a permutation matrix.

4.4.2 Identifiability with partially known $\mathbf{R}^x$

The last subsection assumed that second-order statistics of $\mathbf{x}_t$ were available for all time slots $m = 1, \ldots, M$. However, ample empirical evidence suggests that such information may not be fully available at times. For instance, not all statistics of the stock prices may be available to a given investor in financial markets over time. In brain connectivity studies, one may only have explicit knowledge about exogenous variables in some experimental settings, but not others. Such limitations motivate the analysis of identifiability in settings where one only has access to partial information about second-order statistics of exogenous inputs; that is, $\mathbf{R}^x$ contains misses.

In order to capture the partial availability of $\mathbf{R}^x$, suppose Ω_i denotes set of indices corresponding to known entries per column i of $\mathbf{R}^x$. Furthermore, let $\check{\mathbf{r}}_i^j$ denote a sub-vector of $\mathbf{r}_i^x$, whose entries are indexed by $\Omega_i \cup \Omega_j$ (recall that $\mathbf{r}_i^x$ denotes the i-th column of $\mathbf{R}^x$). Based on these definitions, the next theorem establishes identifiability conditions for settings where $\mathbf{R}^x$ is only partially available.

Theorem 6 *If $\check{\mathbf{r}}_i^j$ and $\check{\mathbf{r}}_j^i$ are linearly independent for any $i \neq j$, then the network adjacency matrix $\mathbf{A}$ can be uniquely identified via Algorithm 4b.*

Proof: Suppose there exists an alternative PARAFAC solution $(\check{\boldsymbol{\mathcal{A}}}, \check{\boldsymbol{\mathcal{A}}}, \check{\mathbf{R}}^x)$ that also decomposes $\underline{\mathbf{R}}^y$ into N rank-one tensors (cf. S2 in Algorithm 4b). According to Theorem 4, there exists a permutation matrix $\check{\mathbf{\Pi}}$ and diagonal scaling matrices $\{\check{\mathbf{\Lambda}}_1, \check{\mathbf{\Lambda}}_2, \check{\mathbf{\Lambda}}_3\}$ such that

$$\check{\mathbf{\Lambda}}_1 \check{\mathbf{\Lambda}}_2 \check{\mathbf{\Lambda}}_3 = \mathbf{I} \tag{4.18}$$

and

$$\check{\boldsymbol{\mathcal{A}}} \;=\; \boldsymbol{\mathcal{A}} \check{\mathbf{\Pi}} \check{\mathbf{\Lambda}}_1 \tag{4.19a}$$

$$\check{\boldsymbol{\mathcal{A}}} = \boldsymbol{\mathcal{A}}\check{\boldsymbol{\Pi}}\check{\boldsymbol{\Lambda}}_2 \tag{4.19b}$$

$$\check{\mathbf{R}}^x = \mathbf{R}^x\check{\boldsymbol{\Pi}}\check{\boldsymbol{\Lambda}}_3 \tag{4.19c}$$

where from (4.14a) and (4.14b), it is clear that $\check{\boldsymbol{\Lambda}}_1 = \check{\boldsymbol{\Lambda}}_2$. On the other hand, when $\mathbf{R}^x$ is partially known; that is, $[\check{\mathbf{R}}^x]_{i,j} = [\mathbf{R}^x]_{i,j}$, for $(i,j) \in \Omega$, then (4.19c) can be written as

$$[\mathbf{R}^x]_{i,j} = [\mathbf{R}^x\check{\boldsymbol{\Pi}}\check{\boldsymbol{\Lambda}}_3]_{i,j}, \qquad \forall\ (i,j) \in \Omega. \tag{4.20}$$

The rest of the proof of Theorem 6 builds on the following lemma.

Lemma 10: *For a given permutation matrix* $\check{\boldsymbol{\Pi}}$*, and scaling matrix* $\check{\boldsymbol{\Lambda}}_3$*, if* $\mathbf{R}^x$ *satisfies the condition in Theorem 5, then* (4.20) *holds true if and only if*

$$\check{\boldsymbol{\Pi}} = \mathbf{I} \tag{4.21a}$$

$$\check{\boldsymbol{\Lambda}}_3 = \mathbf{I}. \tag{4.21b}$$

Proof: See Appendix 4.8.4.

Upon substituting of (4.21a) into (4.19a), and letting $\check{\boldsymbol{\Lambda}} = \check{\boldsymbol{\Lambda}}_1 = \check{\boldsymbol{\Lambda}}_2$, it turns out that

$$\check{\boldsymbol{\mathcal{A}}} = \boldsymbol{\mathcal{A}}\check{\boldsymbol{\Lambda}} \tag{4.22}$$

and the conclusion of Theorem 5 follows from Lemma 9.

Remark 3: The central premise of Theorem 5 is that even when $\mathbf{R}^x$ contains misses, it is possible to uniquely identify the adjacency matrix $\mathbf{A}$. In turn, this facilitates the combination of information pertaining to nodal processes from different time slots towards the task of inference of the hidden network topology, even though complete correlation information is unavailable for all the nodes.

Our novel tensor-based topology identification approach advocated so far focuses on settings where the network topology does not vary with time. The rest of the chapter goes beyond this assumption, and explores scenarios where the link structure may even evolve over time, with the ultimate goal of tracking the network topology, possibly in real time.

4.5 Tracking dynamic network topologies

It has hitherto been taken for granted that all past data are available, and the developed tensor-based approaches will operate in batch mode. In fact, Algorithm 4b is conducted entirely offline, with $\underline{\mathbf{R}}^y$ obtained or computed a priori. However, practical constraints often render it impossible to operate in batch mode; for instance, nodal data in large-scale networks (e.g., modern social media and the web) can only be acquired in real-time streams since any attempts to store such data for batch processing will quickly overwhelm operators.

Equally important is the observation that most real-world networks evolve over time, namely, new edges and nodes may appear, while others become obsolete during the observation period. Consequently, even if a batch approach were to overcome challenges due to the sheer scale of the data, the inferred networks would represent a single aggregate perspective of several evolving network topologies at best. In lieu of these challenges, this section extends the novel tensor-based approach to track changes to the network topologies in real time.

4.5.1 Piecewise-invariant dynamic network topologies

Suppose that the network exhibits a piecewise-constant topology, captured by the sequence of unknown adjacency matrices $\{\mathbf{A}_m \in \mathbb{R}^{N\times N},\ t \in [\tau_m, \tau_{m+1}-1]\}_{m=1}^{M}$, over M time segments. Each entry (i,j) of $\mathbf{A}_m$ is nonzero only if a directed edge exists from node i to j, and it will be denoted by a_{ij}^m. Similarly associating each node with a single exogenous input, one obtains the following SEM

$$y_{jt} = \sum_{i\neq j} a_{ij}^m y_{it} + b_{jj}^m x_{jt} + e_{jt}, \quad t \in [\tau_m, \tau_{m+1}-1] \tag{4.23}$$

per $m = 1,\dots,M$, with e_{jt} similarly capturing unmodeled dynamics, while coefficients $\{a_{ij}^m\}$ and $\{b_{jj}^m\}$ are unknown. With $\mathbf{y}_t$, $\mathbf{x}_t$, and $\mathbf{e}_t$ previously defined, (4.23) can be written in vector form as

$$\mathbf{y}_t = \mathbf{A}_m \mathbf{y}_t + \mathbf{B}_m \mathbf{x}_t + \mathbf{e}_t \tag{4.24}$$

where $[\mathbf{A}_m]_{ij} = a_{ij}^m$ and $\mathbf{B}_m := \text{Diag}(b_{11}^m, \dots, b_{NN}^m)$. Based on (4.24), we will develop an algorithm to track $\{\mathbf{A}_m, \mathbf{B}_m\}_{m=1}^{M}$ using measured endogenous variables, and the sequence of

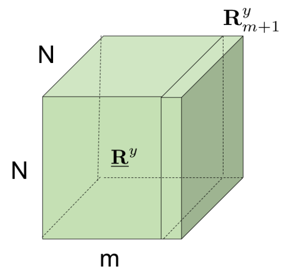


Figure 4.3: Tensor grows per window m by a new frontal slice.

correlation matrices $\{\mathbf{R}^x_m\}_{m=1}^M$.

Key to the novel topology tracking algorithm is recognizing that the tensor-based approach of Section 4.3 can be extended to settings where the network exhibits piecewise-constant topology variations. To this end, define $\boldsymbol{\mathcal{A}}_m := (\mathbf{I} - \mathbf{A}_m)^{-1}\mathbf{B}_m$, and consider a tensor with the m-th slice

$$\mathbf{R}^y_m = \boldsymbol{\mathcal{A}}_m \mathbf{R}^x_m \boldsymbol{\mathcal{A}}_m^\top, \quad t \in [\tau_m, \tau_{m+1} - 1] \tag{4.25}$$

sequentially appended at $t = \tau_{m+1}$, for $m = 1, \ldots, M$; see also (4.5) and Figure 4.3. Allowing $\underline{\mathbf{R}}^y$ to grow sequentially along one mode is well motivated for real-time operation, where data may be acquired in a streaming manner. In this case, unveiling the evolving network topology calls for approaches that are capable of tracking tensor factors. In fact, the topology tracking algorithm developed next builds upon a prior sequential tensor factorization approach, namely, PARAFAC via recursive least-squares tracking (PARAFAC-RLST); see e.g., [125] for details.

4.5.2 Exponentially-weighted least-squares estimator

Let $\bar{\mathbf{r}}^y_m := \text{vec}(\mathbf{R}^y_m)$ denote the vectorization of $\mathbf{R}^y_m$, and note that $\bar{\mathbf{r}}^y_m$ can be written as $\bar{\mathbf{r}}^y_m = \mathbf{H}_m \boldsymbol{\rho}^x_m$, where $\mathbf{H}_m := \boldsymbol{\mathcal{A}}_m \odot \boldsymbol{\mathcal{A}}_m$ is an $N^2 \times N$ matrix, and $\boldsymbol{\rho}^x_m$ is defined after (4.5). To track $\mathbf{H}_m$, we advocate an exponentially-weighted least-squares estimator, namely,

$$\widehat{\mathbf{H}}_m = \arg\min_{\mathbf{H}} \sum_{l=1}^m \beta^{m-l} \|\bar{\mathbf{r}}^y_l - \mathbf{H}\boldsymbol{\rho}^x_l\|_2^2 \tag{4.26}$$

for $m = 1, \ldots, M$, where $\beta \in (0, 1]$ denotes a forgetting factor, which facilitates tracking topology changes by down-weighing past data when $\beta < 1$.

Letting $f_m(\mathbf{H}) := \sum_{l=1}^m \beta^{m-l} \|\bar{\mathbf{r}}^y_l - \mathbf{H}\boldsymbol{\rho}^x_l\|_2^2$ denote the cost function per segment m, and taking the gradient with respect to $\mathbf{H}$, one obtains

$$\nabla f_m(\mathbf{H}) = 2\sum_{l=1}^{m} \beta^{m-l} \left(\bar{\mathbf{r}}_l^y - \mathbf{H}\boldsymbol{\rho}_l^x\right) (\boldsymbol{\rho}_l^x)^\top. \tag{4.27}$$

Setting $\nabla f_m(\mathbf{H}) = \mathbf{0}$, and solving for $\mathbf{H}_m$ yields

$$\mathbf{H}_m = \mathbf{Q}_m \mathbf{P}_m^{-1} \tag{4.28}$$

where $\mathbf{Q}_m := \sum_{l=1}^{m} \beta^{m-l}\bar{\mathbf{r}}_l^y(\boldsymbol{\rho}_l^x)^\top$ and $\mathbf{P}_m := \sum_{l=1}^{m} \beta^{m-l}\boldsymbol{\rho}_l^x(\boldsymbol{\rho}_l^x)^\top$. Further inspection of $\mathbf{P}_m$ and $\mathbf{Q}_m$ reveals that the updates admit recursive forms as follows

$$\mathbf{P}_m := \beta\mathbf{P}_{m-1} + \boldsymbol{\rho}_m^x(\boldsymbol{\rho}_m^x)^\top \tag{4.29}$$

$$\mathbf{Q}_m := \beta\mathbf{Q}_{m-1} + \bar{\mathbf{r}}_m^y(\boldsymbol{\rho}_m^x)^\top. \tag{4.30}$$

Moreover, letting $\mathbf{W}_m := \mathbf{P}_m^{-1}$, one can resort to the matrix inversion lemma to recursively compute inverses as

$$\mathbf{W}_m = \beta^{-1}\left[\mathbf{W}_{m-1} - \frac{\mathbf{W}_{m-1}\boldsymbol{\rho}_m^x(\boldsymbol{\rho}_m^x)^\top\mathbf{W}_{m-1}}{\beta + (\boldsymbol{\rho}_m^x)^\top\mathbf{W}_{m-1}\boldsymbol{\rho}_m^x}\right]. \tag{4.31}$$

It is worth pointing out that the simple recursive updates (4.29) - (4.31) lead to a markedly reduced computational burden, while only requiring fixed memory storage costs.

Once $\mathbf{H}_m$ is estimated, $\boldsymbol{\mathcal{A}}_m := [\boldsymbol{\alpha}_{1m}, \ldots, \boldsymbol{\alpha}_{Nm}]$ can be recovered by recalling that the ith column of $\mathbf{H}_m$ is given by

$$\mathbf{h}_{im} = \boldsymbol{\alpha}_{im} \otimes \boldsymbol{\alpha}_{im} = \text{vec}(\boldsymbol{\alpha}_{im}\boldsymbol{\alpha}_{im}^\top). \tag{4.32}$$

Recognizing that $\bar{\mathbf{H}}_{im} := \boldsymbol{\alpha}_{im}\boldsymbol{\alpha}_{im}^\top$ is a rank one matrix, $\boldsymbol{\alpha}_{im}$ can be estimated via the leading eigenvector of $\bar{\mathbf{H}}_{im}$, namely

$$\widehat{\boldsymbol{\alpha}}_{im} \approx \lambda_{\max}^{\frac{1}{2}}(\bar{\mathbf{H}}_{im})\mathbf{v}_{\max}(\bar{\mathbf{H}}_{im}) \tag{4.33}$$

where the eigen-pair $\{\lambda_{\max}(\bar{\mathbf{H}}_{im}), \mathbf{v}_{\max}(\bar{\mathbf{H}}_{im})\}$ denotes the leading eigenvalue of $\bar{\mathbf{H}}_{im}$, and its corresponding eigenvector, both obtainable via the power iteration [138]. This is carried out per column of $\boldsymbol{\mathcal{A}}_m$ to obtain $\widehat{\boldsymbol{\mathcal{A}}}_m := [\widehat{\boldsymbol{\alpha}}_{1m}, \ldots, \widehat{\boldsymbol{\alpha}}_{Nm}]$, while $\mathbf{A}_m$ can be estimated as (cf.

Algorithm 5b Tensor-based network topology tracking

```
**Input:** $\{\boldsymbol{\rho}_m^x\}_{m=1}^M$, $\{\mathbf{y}_t\}$, $\beta$, $\mathbf{W}_0$, $\mathbf{Q}_0 = \mathbf{0}$, $\eta$
**for** $m = 1, \ldots, M$ **do**
    **S1.** Tensor formation
        Set frontal slice $m$ of $\underline{\mathbf{R}}^y$ to $\widehat{\mathbf{R}}_m^y$ as in (4.8)
    **S2.** Variable updates:
        $\mathbf{Q}_m := \beta\mathbf{Q}_{m-1} + \bar{\mathbf{r}}_m^y(\boldsymbol{\rho}_m^x)^\top$
        Update $\mathbf{W}_m$ via (4.31)
        Uptate $\widehat{\boldsymbol{\alpha}}_{im}$ via (4.33), for $i = 1, \ldots N$
    **S3.** SEM estimates for topology tracking:
        Estimate $\hat{\mathbf{A}}_m$ via (4.34).
    Return $\hat{\mathbf{A}}_m$
**end for**
**Edge identification:**
$[\hat{\mathbf{A}}_m]_{ij} \neq 0$ **if** $[\hat{\mathbf{A}}_m]_{ij} > \eta$**, otherwise** $[\hat{\mathbf{A}}_m]_{ij} = 0$**,** $\forall(i,j)$
```

Proposition 3)

$$\hat{\mathbf{A}}_m = \mathbf{I} - \left(\text{Diag}(\widehat{\boldsymbol{\mathcal{A}}}_m^{-1})\right)^{-1} \widehat{\boldsymbol{\mathcal{A}}}_m^{-1}. \tag{4.34}$$

Algorithm 5b lists the steps involved in tracking evolving network topologies via the scheme advocated in this section.

Remark 4 (Initialization): Matrix $\mathbf{P}_m$ in (4.29) is rank deficient when $m \leq N$, rendering the update in (4.28) impossible. This can be addressed by setting $\mathbf{W}_0 = \mathbf{P}_0^{-1} = a\mathbf{I}$, for a very large constant a (e.g., $a = 10^5$). Since $\mathbf{P}_m^{-1}$ is a variance estimate of $\widehat{\mathbf{H}}_m$, this initialization amounts to placing little confidence in the initial values. Matrix $\mathbf{Q}_0$ is initialized as an all-zero matrix.

4.6 Numerical Tests

In order to assess the effectiveness of the novel algorithms, this section presents test results from experiments conducted on both simulated and real network data. Consideration was given to scenarios involving both static and dynamic networks.

4.6.1 Tests on static simulated networks

Data generation. A *Kronecker random graph* comprising $N = 64$ nodes was generated from a prescribed "seed matrix"

$$\mathbf{S}_0 := \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix}$$

in order to obtain a binary-valued 64×64 matrix via repeated Kronecker products, namely $\mathbf{S} = \mathbf{S}_0 \otimes \mathbf{S}_0 \otimes \mathbf{S}_0$; see also [139]. Using the binary matrix $\mathbf{S}$ to describe the zero and nonzero entries of the topology, the Kronecker graph with adjacency matrix $\mathbf{A}$ was then constructed by randomly sampling each entry from a uniform distribution with $a_{ij} \sim$ Unif$(0.2s_{ij}, 0.5s_{ij})$. To generate synthetic endogenous measurements, the observation horizon was set to $T = ML$ time-slots, which were partitioned into M windows of fixed length L, using pre-selected boundaries $\{\tau_m\}_{m=1}^{M+1}$ with $\tau_1 = 1$ and $L := \tau_{m+1} - \tau_m$, for several values of L and M. Per $t \in [\tau_m, \tau_{m+1} - 1]$, exogenous inputs were sampled as $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \sigma_m^2 \mathbf{I})$, with $\{\sigma_m\}_{m=1}^{M}$ set to M distinct values. With $\mathbf{e}_t$ sampled i.i.d. from $\mathcal{N}(\mathbf{0}, \sigma_e^2 \mathbf{I})$, $\mathbf{y}_t$ was generated using the SEM, that is, $\mathbf{y}_t = (\mathbf{I} - \mathbf{A})^{-1}(\mathbf{B}\mathbf{x}_t + \mathbf{e}_t)$, where $\mathbf{B}$ is a diagonal matrix with $[\mathbf{B}]_{jj}$ drawn uniformly from the interval $[2, 3]$.

In order to conduct PARAFAC decompositions, an implementation in the open source Tensorlab 3.0 toolbox was adopted [140]. Upon running Algorithm 4b, an edge was declared present if the estimate $\hat{a}_{ij}$ was found to exceed a prescribed threshold $\eta = 0.2$. The edge identification error rate (EIER) is tested for multiple experimental settings, which is defined as

$$\text{EIER} := \frac{\|\mathbf{S} - \widehat{\mathbf{S}}\|_0}{N(N-1)} \times 100\% \tag{4.35}$$

with the operator $\| \cdot \|_0$ denoting the number of nonzero entries of its argument. Matrix $\mathbf{S} \in \{0, 1\}^{N \times N}$ denotes the ground-truth binary edge indicator matrix, while $\widehat{\mathbf{S}}$ denotes its estimate obtained by the novel scheme.

Experiments were run for different values of M, and error plots were generated using EIER values averaged over 100 independent runs.

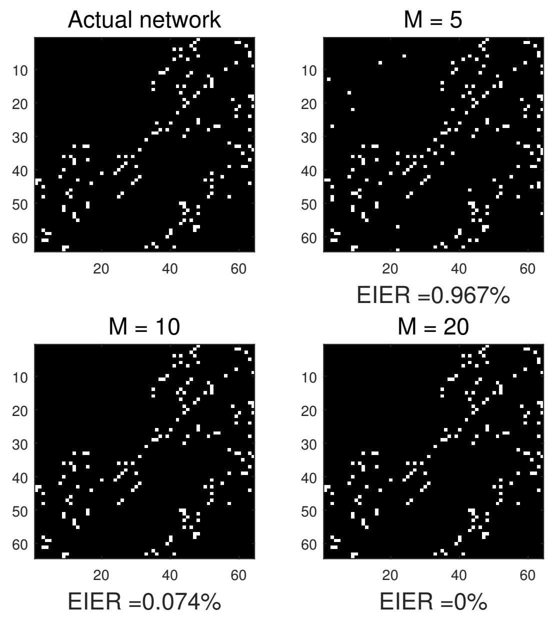


Figure 4.4: Actual and inferred adjacency matrices with the number of windows set to $M = 5, 10$, and 20.

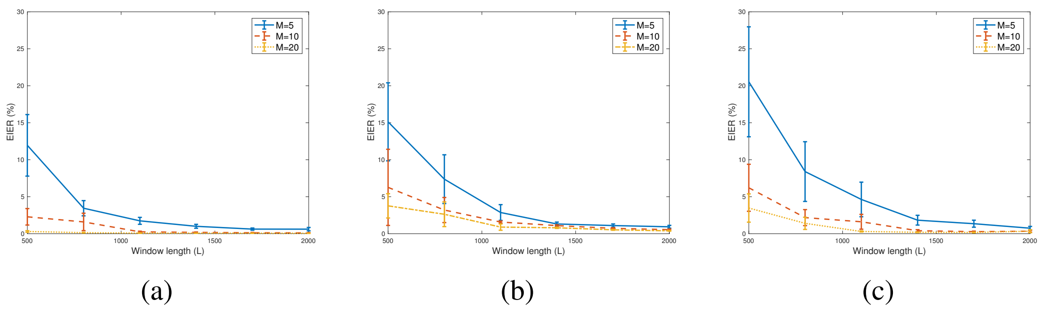
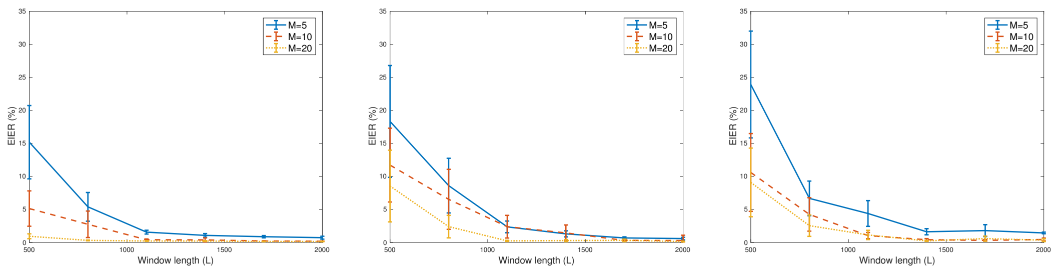


Figure 4.5: EIER under varying window lengths M, and $\sigma_e = 0.1$: a) $\Omega = \{(i,j)|i = 1,\dots,N, j = 1,\dots,M\}$; b) 50% misses in $\mathbf{R}^x$; and c) $\Omega = \emptyset$.

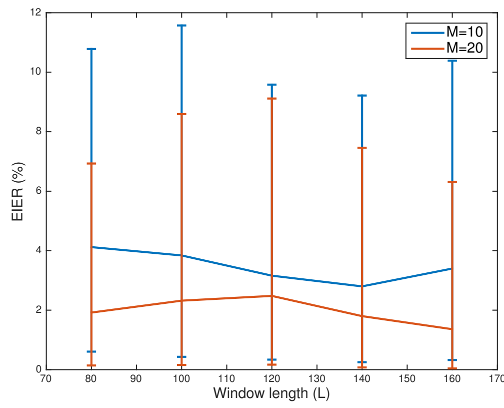
Results. Figure 4.4 depicts actual and inferred adjacency matrices, resulting from one realization of Algorithm 4b for $M \in \{5, 10, 20\}$, with $L = 1,000$ per experiment. As shown in the plot, fewer edges are erroneously identified as the number of windows M increases. This is not really surprising because the probability that the condition in Theorem 5 is satisfied will improve with larger M.



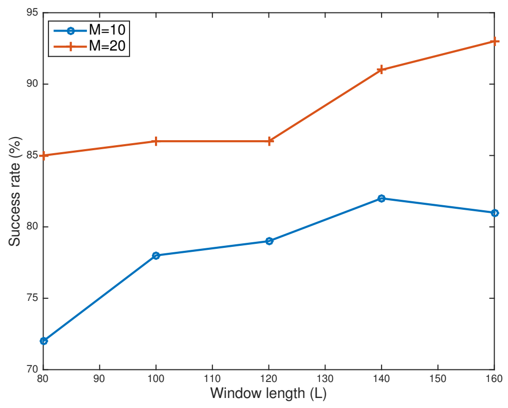
(a) (b) (c)

Figure 4.6: EIER for several values of M and $\sigma_e = 1$: a) $\Omega = \{(i,j)|i = 1,\ldots,N, j = 1,\ldots,M\}$; b) 50% misses in $\mathbf{R}^x$; and c) $\Omega = \emptyset$.

Figures 4.5 and 4.6 plot the observed error performance over several window lengths (L), when noise variance is set to $\sigma_e^2 = 0.01$ and $\sigma_e^2 = 1$, respectively. In both figures, the following settings are considered: (a) $\mathbf{R}^x$ is fully available; (b) random omission of entries in $\mathbf{R}^x$ with probability 0.5; and (c) the completely blind case, that is, $\Omega = \emptyset$. In all three scenarios, there is a general increase in edge identification accuracy with L, since wider window lengths yield improved estimates of the correlation matrices per window. Not surprisingly, the semi-blind topology inference approach in Section 4.4-B outperforms the completely blind alternative ($\Omega = \emptyset$), since one presumably has more prior information available. On the other hand, in the completely blind case, Algorithm 4b still results in a reliable estimate of the network topology with low edge identification error. In several real-world applications, exogenous variables are often unavailable or costly to measure, hence performance benchmarks for the developed algorithm in such blind settings are of considerable interest. To facilitate further assessment of the stability of the novel algorithm when operating in blind scenarios, an extended experiment was carried out as follows. Per experiment trial, an unweighted Erdös-Renyi random graph with 5 nodes was generated, with the probability that any node pair is connected by an edge set to 0.4, and then Algorithm 4b was run with $\Omega = \emptyset$. For this experiment, Figure 4.7 (a) depicts the resulting EIER performance, averaged over 100 independent runs. Figure 4.7 (b) depicts the success rate of the experiments, with a trial is considered successful if EIER $= 0$. It is clear from the results that the majority of trials succeeded in exact identification of all edges. This is an exciting empirical result that demonstrates the potential for the proposed algorithm to provide reliable estimates in blind settings, even under the presence of noise. The implications of this empirical result are well-motivated in real-world applications, where exogenous inputs are



(a)



(b)

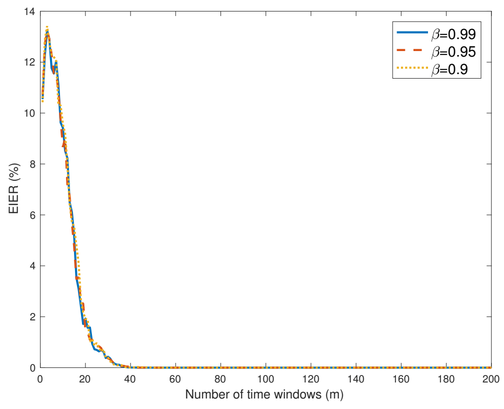
Figure 4.7: Performance in blind scenario: a) EIER; b) Success rate.

unavailable to eliminate the inherent permutation ambiguity.

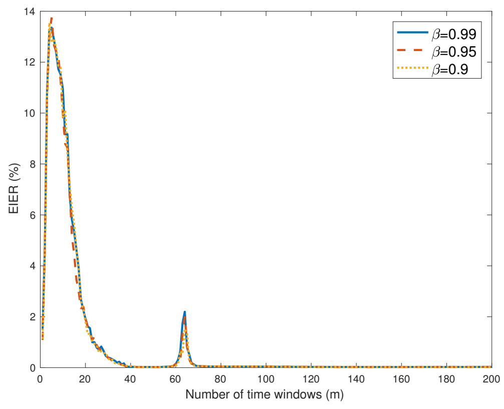
4.6.2 Simulated piecewise-constant network

Data generation. An initial 64-node network was generated with adjacency matrix $\mathbf{A}_0$ via the Kronecker random graph model, as detailed in the previous subsection. Edge weights in the initial non-zero support of $\mathbf{A}_0$ were varied over time windows, following two edge-variation patterns: p1) $a_{ij}^m = a_{ij}^0 + 0.1\sin(0.01m)$, for $m = 1, \ldots, 200$; and p2) $a_{ij}^m = 0$ with probability 0.2 at the 50th and 100th time windows. For $L = 500$, $L = 2,000$, and $L = 3,000$, endogenous measurements were simulated over $T = ML$ time-slots, partitioned into M windows of fixed length L. The window boundaries were preselected as $\{\tau_m\}_{m=1}^{M+1}$, with $\tau_1 = 1$ and $L := \tau_{m+1} - \tau_m$. Per $t \in [\tau_m, \tau_{m+1} - 1]$, exogenous inputs were sampled as $\mathbf{x}_t \sim \mathcal{N}(\mathbf{0}, \sigma_m^2 \mathbf{I})$, with $\{\sigma_m^2\}_{m=1}^M$ set to M distinct values. With $\mathbf{e}_t$ sampled i.i.d. from $\mathcal{N}(\mathbf{0}, 10^{-2}\mathbf{I})$, $\mathbf{y}_t$ was similarly generated using the SEM, that is, $\mathbf{y}_t = (\mathbf{I} - \mathbf{A}_m)^{-1}(\mathbf{B}\mathbf{x}_t + \mathbf{e}_t)$, where $[\mathbf{B}]_{jj} \sim \mathcal{U}[2, 3]$.

Results. Algorithm 5b was run on the simulated data using $\beta = \{0.99, 0.95, 0.9\}$, with an edge declared present if $\hat{a}_{ij}$ exceeded a threshold $\eta = 0.2$. Algorithm performance was assessed with respect to both EIER, and the empirical mean-square error (E-MSE), defined as E-MSE $:= \|\mathbf{A}_m - \hat{\mathbf{A}}_m\|_F^2/(N(N-1))$. In addition, both error metrics were averaged over 100 runs per experiment. It is generally observed that tracking performance is not very sensitive to the selection of β.

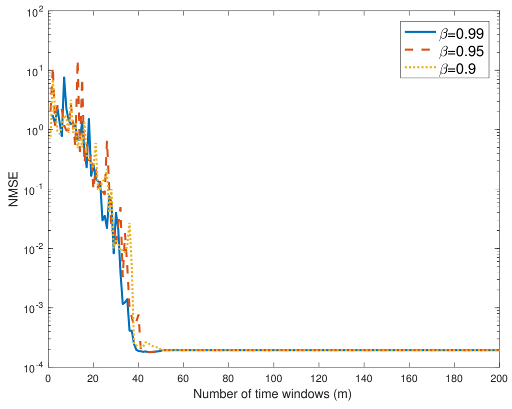


(a)

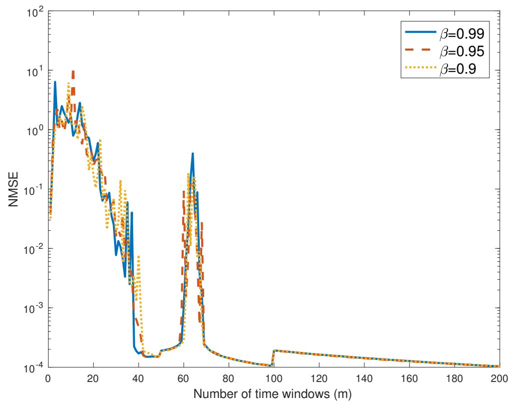


(b)

Figure 4.8: EIER vs. m for: (a) Scenario p1; and (b) Scenario p2.



(a)



(b)

Figure 4.9: MSE vs m for: a) Scenario p1; b) Scenario p2.

As shown by both Figures 4.8 and 4.9, Algorithm 5b tracks the evolution of the network remarkably well. During windows where the edge support is known to change, error metrics increase in value, but gracefully return to lower values. Figure 4.10 depicts heatmaps of actual and inferred adjacency matrices, obtained by running Algorithm 5b during the window indexed by $m = 200$ for scenario p2).

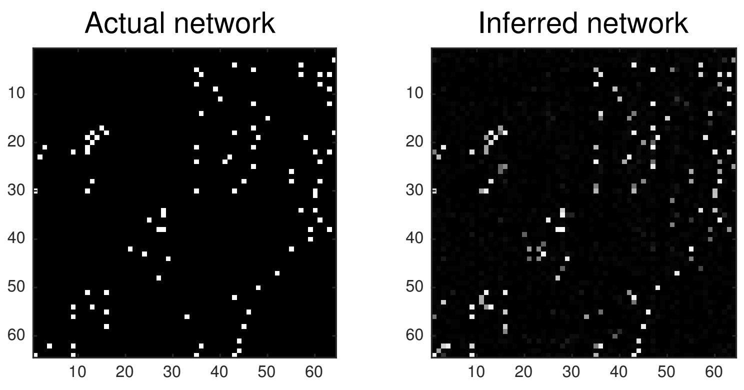


Figure 4.10: Actual and inferred networks at $m = 200$.

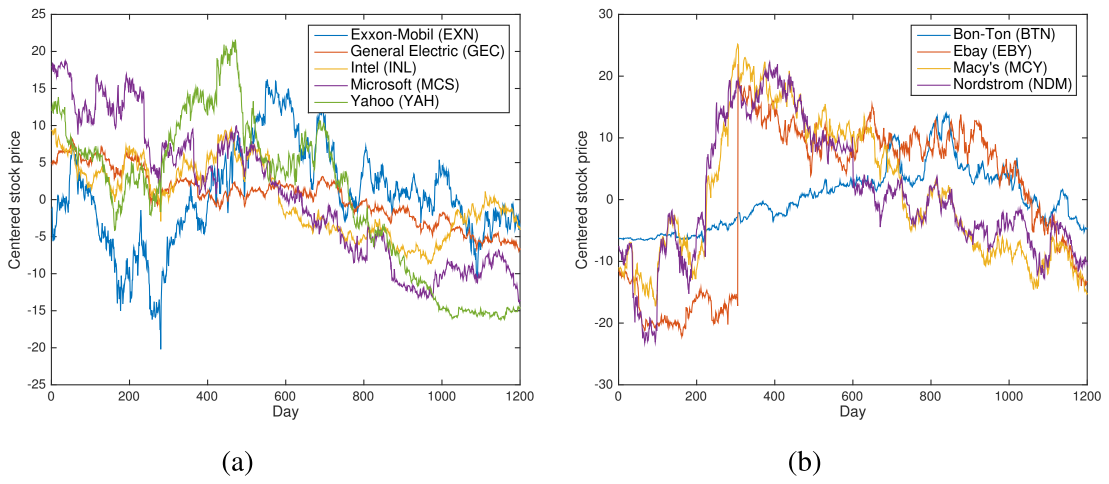
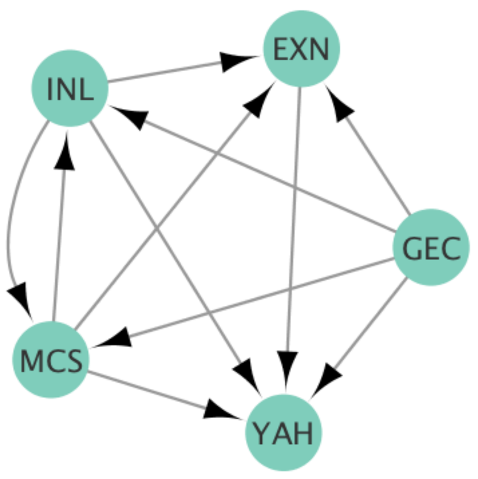


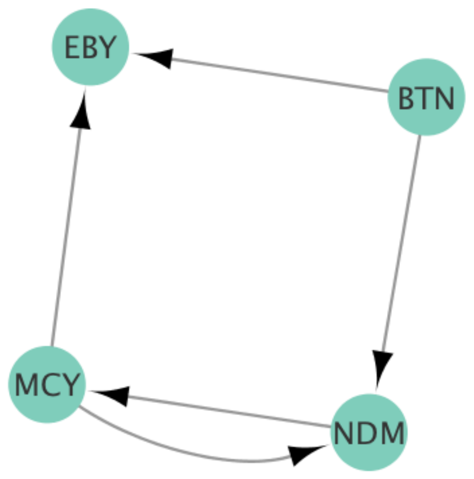
Figure 4.11: Plot of the two groups of stock prices over the observation duration with zero-mean centering: a) technology companies; and b) online and "brick-and-mortar" retailers. The stock ticker symbol for each company is shown in the legend (in parentheses).

4.6.3 Tests on real networks

Data description. To conduct tests on real-world networks, historical stock price data were downloaded through a free *Yahoo* application program interface (API) [141]. Historical closing prices were obtained as time series for dates ranging from December 23, 2011 to September 30, 2016 ($1,200$ days in total). The stock time series were grouped into two clusters, namely: a) large technology companies (*Exxon-Mobil, Intel, Microsoft, Yahoo, and General Electric*), and b) online and brick-and-mortar retailers (*Bon-Ton, E-bay, Macy's, and Nordstrom*). Choices of



(a)



(b)

Figure 4.12: Visualization of network topologies inferred from the stock price time series, depicting: a) technology companies; and b) online and "brick-and-mortar" retailers. Notice the stronger dependencies between the two competing "brick-and-mortar" retailers, Macy's (MCY) and Nordstrom (NDM).

which stocks were classified under the two groups were based on prior knowledge of historical inter-dependencies existing among them in financial markets. For instance, a significant drop in Intel stock prices often signals changes in share prices for Microsoft, Intel, and sometimes General Electric.

Results. For this set of experiments, the combined multivariate time series were adopted as endogenous variables $\left(\{\mathbf{y}_t\}_{t=1}^{1,200}\right)$, after a pre-processing step in which they were centered to have zero mean; see Figure 4.11 for a plot of the centered time series. Furthermore, money invested in the stocks constitutes exogenous inputs $\left(\{\mathbf{x}_t\}_{t=1}^{1,200}\right)$, which are not known in this case, since such information is generally not privy to the public, hence $\Omega = \emptyset$. Furthermore, it was observed that most stock prices tend to exhibit steady quarterly trends (rising or falling), and the window length was consequently set to $L \in \{60, 80, 100\}$. Algorithm 4b was then run with $\Omega = \emptyset$, to infer the causal dependencies between the selected stock prices.

According to the discussion in Section 4.4, there is no guarantee of identifiability in the completely blind setting. Fortunately, the simulated tests depicted by Figure 4.7 demonstrate that when the network has a few nodes, there is a high probability of successful recovery of the

Data segment	Network 1	Network 2
$t \in \{1, \ldots, 1,200\}$	92	68
$t \in \{1, \ldots, 1,000\}$	86	86
$t \in \{1, \ldots, 800\}$	78	60
$t \in \{201, \ldots, 1,200\}$	56	92

Table 4.1: Frequency of inference of networks depicted in Figure 4.12 out of 100 independent runs of Algorithm 5b on different segments of data with window lengths $L = 100$.

true network in the presence of noise. Based on this empirical observation, it is reasonable to expect that if only a few stocks are selected, then many trials will yield the true network upon running Algorithm 4b with random initializations. To this end, 100 independent runs of Algorithm $4b$ were conducted with random initializations, and the edge detection threshold was set to $0.1 \max_{ij} |\hat{a}_{ij}|$. This is admittedly an ad hoc threshold, and investigation of more sophisticated approaches is possible from e.g., [142]. It turned out that most estimates yielded the same support for $\hat{\mathbf{A}}$, with very slight variations in actual values of its entries. Consequently, a simple scheme was adopted to infer the network topology from the ensemble of estimates. Unique topologies based on the support of $\hat{\mathbf{A}}$ for the 100 realizations were enumerated, and a majority voting scheme was adopted to reach consensus on the final topology. The most frequent network topologies from the experiments are depicted by Figure 4.12. The figure shows very strong dependencies in the first group of technology companies, while the second plot shows stronger inter-dependencies between Macy's and Nordstrom than the others. Interestingly, both Macy's and Nordstrom are well-known "brick-and-mortar" retailers and competitors. The stronger dependence between them seems to agree with the expectation that changes in the price of one would be expected to indirectly impact the other.

Furthermore, Table 4.1 lists the frequency of appearance (out of 100 independent trials) of the networks depicted in Figure 4.12 for varying data segments, upon running Algorithm 5b. Similarly, the same frequencies resulting from running Algorithm 4b with different window lengths on the entire dataset are shown in Table 4.2. It is clear from these tables that the same network topologies are inferred with high probability in most of the cases.

Window length (L)	Network 1	Network 2
100	92	68
80	89	71
60	53	55

Table 4.2: Frequency of inference of networks depicted in Figure 4.12 out of 100 independent runs of Algorithm 5b on the entire data set with different window lengths.

4.7 Summary

This chapter put forth a novel approach for inference of network topologies from the statistics of nodal processes. Leveraging SEMs, the network topology inference task was reformulated as a constrained PARAFAC tensor decomposition. Recognizing the inherent uniqueness challenges, conditions under which the network can be uniquely identified were derived. Unlike conventional SEMs, which require exact information of the exogenous inputs in order to guarantee identifiability, it was proven that the novel tensor-based approach is capable of uniquely identifying the network topology only with partial information of the second-order statistics of nodal exogenous inputs.

The framework was further extended to facilitate real-time sequential estimation of the network topology by developing a novel topology tracking algorithm. An exponentially weighted least-squares estimator was advocated for the topology tracking problem, making it possible to efficiently solve the problem "on the fly." To assess the effectiveness of the novel approaches, extensive numerical tests were conducted on both simulated data and historical stock prices of several publicly-traded corporations.

In order to broaden the scope of this work, there are several intriguing directions for future investigation, namely: a) developing algorithms that are capable of exploiting prior knowledge pertaining to the network structure e.g., edge sparsity or power law degree distributions; and b) distributed implementation of the novel algorithms, which is well-motivated, especially when dealing with large-scale networks.

4.8 Appendix

4.8.1 Proof of Proposition 3

Since diagonal entries of $\mathbf{A}$ are all zero, and $\mathbf{B}^{-1}$ is a diagonal matrix with nonzero entries, $\boldsymbol{\mathcal{A}}$ is invertible; that is,

$$\boldsymbol{\mathcal{A}}^{-1} = \mathbf{B}^{-1}(\mathbf{I} - \mathbf{A}). \tag{4.36}$$

Clearly, the diagonal entries of $\boldsymbol{\mathcal{A}}^{-1}$ coincide with those of $\mathbf{B}^{-1}$, which implies that

$$\mathbf{B} = \left(\text{Diag}\left[\boldsymbol{\mathcal{A}}^{-1}\right]\right)^{-1}. \tag{4.37}$$

Recognizing that $\mathbf{B}\boldsymbol{\mathcal{A}}^{-1} = \mathbf{I} - \mathbf{A}$, one can write

$$\mathbf{A} = \mathbf{I} - \mathbf{B}\boldsymbol{\mathcal{A}}^{-1} = \mathbf{I} - \left(\text{Diag}(\boldsymbol{\mathcal{A}}^{-1})\right)^{-1}\boldsymbol{\mathcal{A}}^{-1} \tag{4.38}$$

which completes the proof.

4.8.2 Proof of Lemma 8

First, note that (4.15) can be written as

$$\mathbf{R}^x - \mathbf{R}^x\boldsymbol{\Pi}\boldsymbol{\Lambda}_3 = \mathbf{0}_{M\times N} \tag{4.39}$$

and recall that $\boldsymbol{\Pi}$ is a permutation matrix; hence, each constituent column in $\boldsymbol{\Pi}$ comprises zeros with the exception of a single entry set to one. Letting π_{ij} denote the (i,j)-th entry of $\boldsymbol{\Pi}$, assume without loss of generality that $\pi_{ij} = 1$ and $\pi_{kj} = 0$, $\forall k \neq i$. Consequently, with $\mathbf{p}_j \in \mathbb{R}^N$ representing column j of $\mathbf{P} := \boldsymbol{\Pi}\boldsymbol{\Lambda}_3$, one can equivalently write

$$\mathbf{p}_j = [0, \ldots, 0, \underbrace{\pi_{ij}\lambda_j}_{\text{entry } i}, 0, \ldots, 0]^\top \tag{4.40}$$

where $\lambda_j \neq 0$ denotes the j-th diagonal entry of $\mathbf{\Lambda}_3$. Extracting the j-th column on both sides of (4.39), namely,

$$\mathbf{r}_j^x - \mathbf{R}^x \mathbf{p}_j = \mathbf{0}_{M\times 1} \tag{4.41}$$

and combining (4.40) and (4.41), one obtains

$$\mathbf{r}_j^x = \pi_{ij}\lambda_j \mathbf{r}_i^x. \tag{4.42}$$

When $i \neq j$, (4.42) implies that $\mathbf{r}_i^x$ and $\mathbf{r}_j^x$ are linearly dependent, which contradicts the condition $\text{kr}(\mathbf{R}^x) \geq 2$ in Lemma 8. Hence, for (4.42) to hold for some nonzero λ_j, it is necessary that $i = j$, which is equivalent to requiring $\pi_{jj} = 1$ and $\lambda_j = 1$. Since this holds for any j, one deduces that

$$\mathbf{\Pi} = \mathbf{I}, \;\; \mathbf{\Lambda}_3 = \mathbf{I}. \tag{4.43}$$

4.8.3 Proof of Lemma 9

Recalling from Algorithm 4b that

$$\hat{\mathbf{A}} \;\; = \;\; \mathbf{I} - \left(\text{Diag}(\widehat{\boldsymbol{\mathcal{A}}}^{-1})\right)^{-1} \widehat{\boldsymbol{\mathcal{A}}}^{-1}$$

and substituting $\widehat{\boldsymbol{\mathcal{A}}} = \boldsymbol{\mathcal{A}}\mathbf{\Lambda}$, one obtains

$$\begin{aligned} \hat{\mathbf{A}} &= \mathbf{I} - \left(\text{Diag}\left[(\boldsymbol{\mathcal{A}}\mathbf{\Lambda})^{-1}\right]\right)^{-1} (\boldsymbol{\mathcal{A}}\mathbf{\Lambda})^{-1} \\ &= \mathbf{I} - \left(\text{Diag}\left[(\boldsymbol{\mathcal{A}})^{-1}\right]\right)^{-1} \mathbf{\Lambda}\mathbf{\Lambda}^{-1}\boldsymbol{\mathcal{A}}^{-1} \\ &= \mathbf{I} - \left(\text{Diag}(\boldsymbol{\mathcal{A}}^{-1})\right)^{-1} \boldsymbol{\mathcal{A}}^{-1}. \end{aligned} \tag{4.44}$$

Comparing with Proposition 3, it is clear that $\hat{\mathbf{A}} = \mathbf{A}$, which concludes the proof.

4.8.4 Proof of Lemma 10

First, assume without loss of generality that column j of the permutation matrix $\check{\mathbf{\Pi}}$ satisfies $\check{\pi}_{ij} = 1$ and $\check{\pi}_{kj} = 0$, $\forall k \neq i$, with $\check{\pi}_{ij}$ denoting entry (i, j) of $\check{\mathbf{\Pi}}$. Since $\check{\mathbf{p}}_j \in \mathbb{R}^N$, the j-th column of $\check{\mathbf{P}} := \check{\mathbf{\Pi}}\check{\mathbf{\Lambda}}_3$ can be written as

$$\check{\mathbf{p}}_j := [0, \ldots 0, \underbrace{\check{\pi}_{ij}\check{\lambda}_j}_{\text{entry } i}, 0, \ldots, 0]^\top \tag{4.45}$$

with $\lambda_j \neq 0$ representing the j-th diagonal entry of $\mathbf{\Lambda}_3$. Extracting entries indexed by $\Omega_i \cup \Omega_j$ in column j on both sides of (4.20), one has

$$\check{\mathbf{r}}_j^i = \check{\pi}_{ij}\check{\lambda}_j\check{\mathbf{r}}_i^j \tag{4.46}$$

and assuming that $i \neq j$, (4.46) implies that $\check{\mathbf{r}}_i^j$ and $\check{\mathbf{r}}_j^i$ are linearly dependent, which contradicts the condition in Theorem 5. As a result, for (4.46) to hold true for some nonzero λ_j, it is necessary that $i = j$, which is equivalent to having $\check{\pi}_{jj} = 1$ and $\check{\lambda}_j = 1$. Recognizing that this holds for any j, one arrives at

$$\check{\mathbf{\Pi}} = \mathbf{I}, \;\; \check{\mathbf{\Lambda}}_3 = \mathbf{I}. \tag{4.47}$$

Chapter 5

Nonlinear network topology identification

5.1 Introduction

Several contemporary studies in the neurosciences have converged on the well-accepted view that information processing capabilities of the brain are facilitated by the existence of a complex underlying network; see e.g., [107] for a comprehensive review. The general hope is that understanding the behavior of the brain through the lens of network science will reveal important insights, with an enduring impact on applications in both clinical and cognitive neuroscience.

However, brain networks are not directly observable, and must be inferred from processes observed or measured at nodes. To this end, functional magnetic resonance imaging (fMRI) has emerged as a powerful tool, capable of revealing varying blood oxygenation patterns modulated by brain activity [143]. Other related brain imaging modalities include positron emission tomography (PET), electroencephalography (EEG), and electrocorticography (ECoG), to name just a few. Most state-of-the-art tools for inference of brain connectivity leverage variants of causal and correlational analysis methods, applied to time-series obtained from the imaging modalities [144–148].

Contemporary brain connectivity analyses fall under two broad categories, namely, *functional connectivity* which pertains to discovery of non-directional pairwise correlations between regions of interest (ROIs), and *effective connectivity* which instead focuses on inference of directional (a.k.a., causal) dependencies between them [149]. Granger causality [150], vector autoregressive models (VARMs) [146], structural equation models (SEMs) [151], and dynamic causal modeling (DCM) [152] constitute widely used approaches for effective connectivity studies. VARMs postulate that connected ROIs exert time-lagged dependencies among one another, while SEMs assume instantaneous causal interactions among them. Interestingly, these points of view are unified through the so-termed structural vector autoregressive model (SVARM) [153], which postulates that the spatio-temporal behavior observed in brain imaging data results from both instantaneous and time-lagged interactions between ROIs. It has been shown that SVARMs lead to markedly more flexibility and explanatory power than VARMs and SEMs treated separately, at the expense of increased model complexity [153].

The fundamental appeal of the aforementioned effective connectivity approaches stems from their inherent simplicity, since they adopt linear models. However, this is an oversimplification that is highly motivated by the need for tractability, even though consideration of nonlinear models for causal dependence may lead to more accurate approaches for inference of brain connectivity. In fact, recognizing the limitations associated with linear models, several variants of nonlinear SEMs have been put forth in a number of recent works; see e.g., [154–159].

For example, [157] and [160] advocate SEMs in which nonlinear dependencies only appear among the so-termed *exogenous* variables. Furthermore, [156] puts forth a hierarchical Bayesian nonlinear modeling approach in which unknown random parameters capture the strength and directions of causal links among variables. Several other studies adopt polynomial SEMs, which offer an immediate extension to classical linear SEMs; see e.g., [154, 155, 158, 159]. In all these contemporary approaches, it is assumed that the network connectivity structure is *known a priori*, and developed algorithms only estimate the unknown edge weights. However, this is a rather major limitation since such prior information may not be available in practice, especially when dealing with potentially massive networks, e.g., the brain.

Similarly, several variants of nonlinear VARMs have been shown useful in unveiling links that often remain undiscovered by traditional linear models; see e.g., [161–164]. More recently, [164] proposed a kernel-based VARM, with nonlinear dependencies among nodes encoded by unknown functions belonging to a reproducing kernel Hilbert space.

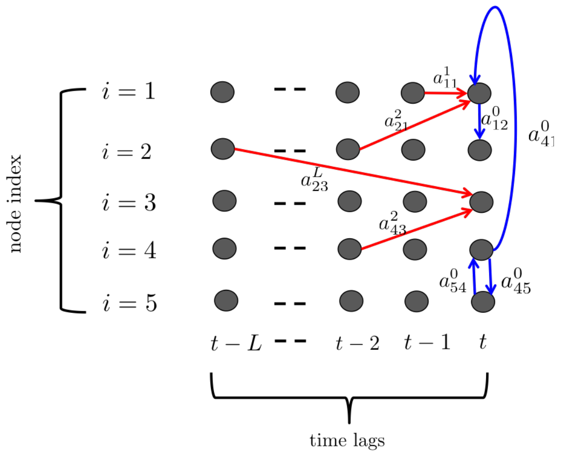
Building upon these prior works, the present chapter puts forth a novel additive nonlinear VARM to capture dependencies between observed ROI-based time-series, without explicit knowledge of the edge structure. Similar to [25, 28], kernels are advocated as an encompassing framework for nonlinear learning tasks. Note that SVARMs admit an interesting interpretation as SEMs, with instantaneous terms viewed as *endogenous* variables, and time-lagged terms as exogenous variables. Since numerical measurement of external brain stimuli is often impractical, or extremely challenging in conventional experiments, adoption of such a fully-fledged SEM (with both endo- and exogenous inputs) is often impossible with traditional imaging modalities.

A key feature of the novel approach is the premise that edges in the unknown network are sparse, that is, each ROI is linked to only a small subset of all potential ROIs that would constitute a maximally-connected power graph. This sparse edge connectivity has recently motivated the development of efficient regularized estimators, promoting the inference of sparse network adjacency matrices; see e.g., [112, 127, 164–167] and references therein. Based on these prior works, this chapter develops a sparse-regularized kernel-based nonlinear SVARM to estimate the effective brain connectivity from per-ROI time series. Compared with [164], the novel approach incorporates instantaneous variables, turns out to be more computationally efficient, and facilitates a data-driven approach for kernel selection.

The rest of this chapter is organized as follows. Section 5.2 introduces the conventional SVARM, while Section 5.3 puts forth its novel nonlinear variant. Section 5.4 advocates a sparsity-promoting regularized least-squares estimator for topology inference from the nonlinear SVARM, while Section 5.5 deals with an approach to learn the kernel that 'best' matches the data. Results of extensive numerical tests based on EEG data from an Epilepsy study are presented in Section 5.6, and pertinent comparisons with linear variants demonstrate the efficacy of the novel approach. Finally, Section 5.7 concludes the chapter, and highlights several potential future research directions opened up by this work.

5.2 Preliminaries on Linear SVARMs

Consider a directed network whose topology is unknown, comprising N nodes, each associated with an observable time series $\{y_{it}\}_{t=1}^{T}$ measured over T time-slots, for $i = 1, \ldots, N$. Note that y_{it} denotes the t-th sample of the time series measured at node i. In the context of the brain, each node could represent a ROI, while the per-ROI time series are obtainable from standard imaging modalities, e.g., EEG or fMRI time courses. The network topology or edge structure will be captured by the weighted graph adjacency matrix $\mathbf{A} \in \mathbb{R}^{N \times N}$, whose (i, j)-th entry a_{ij}



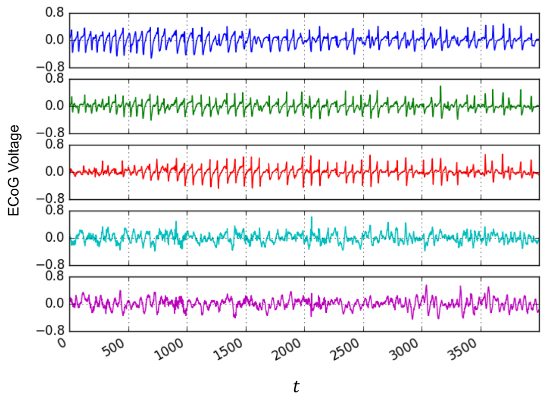


Figure 5.1: (left) A simple illustration of a 5-node brain network; and (right) a set of five neuronal time series (e.g., ECoG voltage) each associated with a node. Per interval t, SVARMs postulate that causal dependencies between the 5 nodal time series may be due to both the instantaneous effects (blue links), and/or time-lagged effects (red links). Estimating the values of the unknown coefficients amounts to learning the causal (link) structure of the network.

is nonzero only if a directed (causal) effect exists from region i to region j.

In order to unveil the hidden causal network topology, traditional linear SVARMs postulate that each y_{jt} can be represented as a linear combination of instantaneous measurements at other nodes $\{y_{it}\}_{i \neq j}$, and their time-lagged versions $\{\{y_{i(t-\ell)}\}_{i=1}^{N}\}_{\ell=1}^{L}$ [153]. Specifically, y_{jt} admits the following linear instantaneous plus time-lagged model

$$y_{jt} = \sum_{i \neq j} a^0_{ij} y_{it} + \sum_{i=1}^{N} \sum_{\ell=1}^{L} a^\ell_{ij} y_{j(t-\ell)} + e_{jt} \tag{5.1}$$

with a^ℓ_{ij} capturing the causal influence of region i upon region j over a lag of ℓ time points, while a^0_{ij} encodes the corresponding instantaneous causal relationship between them. The coefficients encode the causal structure of the network, that is, a causal link exists between nodes i and j only if $a^0_{ij} \neq 0$, or if there exists $a^\ell_{ij} \neq 0$ for $\ell = 1, \ldots, L$. If $a^0_{ij} = 0 \ \forall i, j$, then (5.1) reduces to classical Granger causality [150]. Similarly, setting $a^\ell_{ij} = 0 \ \forall i, j, \ell \neq 0$ reduces (5.1) to a linear SEM with no exogenous inputs [133]. Defining $\mathbf{y}_t := [y_{1t}, \ldots, y_{Nt}]^\top$, $\mathbf{e}_t := [e_{1t}, \ldots, e_{Nt}]^\top$, and the time-lagged adjacency matrix $\mathbf{A}^\ell \in \mathbb{R}^{N \times N}$ with the (i,j)-th entry $\left[\mathbf{A}^\ell\right]_{ij} := a^\ell_{ij}$, one can write (5.1) in vector form as

$$\mathbf{y}_t = \mathbf{A}^0\mathbf{y}_t + \sum_{\ell=1}^{L} \mathbf{A}^\ell \mathbf{y}_{t-\ell} + \mathbf{e}_t \tag{5.2}$$

where $\mathbf{A}^0$ has zero diagonal entries $a_{ii}^0 = 0$ for $i = 1, \ldots, N$.

Given the multivariate time series $\{\mathbf{y}_t\}_{t=1}^T$, the goal is to estimate matrices $\{\mathbf{A}^\ell\}_{\ell=0}^L$, and consequently unveil the hidden network topology. Admittedly, overfitting is a potential risk since L is assumed prescribed. Nevertheless, this can be mitigated via standard order selection methods that control model complexity, e.g., the Bayesian information criterion [168], or Akaike's information criterion [169].

Knowing which entries of $\mathbf{A}^0$ are nonzero, several approaches have been put forth to estimate their values. Examples are based upon ordinary least-squares [153], and hypothesis tests developed to detect presence or absence of pairwise causal links under prescribed false-alarm rates [150]. Albeit conceptually simple and computationally tractable, the linear SVARM is incapable of capturing nonlinear dependencies inherent to complex networks such as the human brain. To this end, the present chapter generalizes the *linear* SVARM in (5.1) to a *nonlinear* kernel-based SVARM.

It is also worth noting that most real world networks (including the brain) exhibit edge sparsity, the tendency for each node to link with only a few other nodes compared to the maximal $\mathcal{O}(N)$ set of potential connections per node. This means that per j, only a few coefficients $\{a_{ij}^\ell\}$ are nonzero. In fact, several recent approaches exploiting edge sparsity have been advocated, leading to more efficient topology estimation; see e.g., [112, 127, 164].

5.3 From linear to nonlinear SVARMs

To enhance flexibility and accuracy, this section generalizes (5.1) so that nonlinear causal dependencies can be captured. The most general nonlinear model with both the instantaneous and time-lagged structure can be written in multivariate form as $\mathbf{y}_t = \bar{\mathbf{f}}(\mathbf{y}_{-jt}, \{\mathbf{y}_{t-\ell}\}_{\ell=1}^L) + \mathbf{e}_t$, or, entry-wise as

$$y_{jt} = \bar{f}_j(\mathbf{y}_{-jt}, \{\mathbf{y}_{t-\ell}\}_{\ell=1}^L) + e_{jt}, \quad j = 1, \ldots, N \tag{5.3}$$

where $\mathbf{y}_{-jt} := [y_{1t}, \ldots, y_{(j-1)t}, y_{(j+1)t}, \ldots, y_{Nt}]^\top$ collects all but the j-th nodal observation at time t, $\mathbf{y}_{t-\ell} := [y_{1(t-\ell)}, \ldots, y_{N(t-\ell)}]^\top$, and $\bar{f}_j(.)$ denotes a nonlinear function of its multivariate argument. With limited (NT) data available, $\bar{f}_j$ in (5.3) entails $(L+1)N-1$ variables. This fact motivates simpler model functions to cope with the emerging 'curse of dimensionality' in estimating $\{\bar{f}_j\}_{j=1}^N$. A simplified form of (5.3) has been studied in [164] with $L = 1$, and without instantaneous influences $\mathbf{y}_{-jt}$, which have been shown of importance in applications such as brain connectivity [151] and gene regulatory networks [113]. Such a model is simplified compared with (5.3) because the number of variables of $\bar{f}_j$ reduces to N. Nevertheless, estimating such an N-variate functional model still suffers from the curse of dimensionality, especially when the size of typical networks scales up.

To circumvent this challenge, we further posit that the multivariate function in (5.3) is separable with respect to each of its $(L+1)N-1$ variables. Such a simplification of (5.3) amounts to adopting a generalized additive model (GAM) [47, Ch. 9]. In the present context, the GAM adopted is $\bar{f}_j(\mathbf{y}_{-jt}, \{\mathbf{y}_{t-\ell}\}_{\ell=1}^L) = \sum_{i \neq j} \bar{f}_{ij}^0(y_{it}) + \sum_{i=1}^N \sum_{\ell=1}^L \bar{f}_{ij}^\ell(y_{i(t-\ell)})$, where the nonlinear functions $\{\bar{f}_{ij}^\ell\}$ will be specified in the next section. Defining $\bar{f}_{ij}^\ell(y) := a_{ij}^\ell f_{ij}^\ell(y)$, the node j observation at time t is a result of both instantaneous and multi-lag effects; that is [cf. (5.1)]

$$y_{jt} = \sum_{i \neq j} a_{ij}^0 f_{ij}^0(y_{it}) + \sum_{i=1}^N \sum_{\ell=1}^L a_{ij}^\ell f_{ij}^\ell(y_{i(t-\ell)}) + e_{jt} \tag{5.4}$$

where similar to (5.1), $\{a_{ij}^\ell\}$ define the matrices $\{\mathbf{A}^\ell\}_{\ell=0}^L$. As before, a directed edge from node j to node i exists if the corresponding $a_{ij}^\ell \neq 0$ for any $\ell = 0, 1, \ldots, L$. Instead of having to estimate an $[(L+1)N-1]$-variate function in (5.3) or an N-variate function in [164], (5.4) requires estimating $(L+1)N-1$ *univariate* functions. Note that conventional linear SVARMs in (5.1) assume that the functions $\{f_{ij}^\ell\}$ in (5.4) are linear, a limitation that the ensuing Section 5.4 will address by resorting to a reproducing kernel Hilbert space (RKHS) formulation to model $\{f_{ij}^\ell\}$.

Problem statement. Given $\{\mathbf{y}_t \in \mathbb{R}^N\}_{t=1}^T$, the goal now becomes to estimate the nonlinear functions $\{f_{ij}^\ell\}$, as well as the adjacency matrices $\{\mathbf{A}^\ell\}_{\ell=0}^L$ in (5.4).

5.4 Kernel-based Sparse SVARMs

Suppose that each univariate function $f_{ij}^{\ell}(.)$ in (5.4) belongs to the RKHS

$$\mathcal{H}_i^{\ell} := \{f_{ij}^{\ell} | f_{ij}^{\ell}(y) = \sum_{t=1}^{\infty} \beta_{ijt}^{\ell} \kappa_i^{\ell}(y, y_{i(t-\ell)})\} \tag{5.5}$$

where $\kappa_i^{\ell}(y, \psi) : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R}$ is a preselected basis (so-termed kernel) function that measures the similarity between y and ψ. Different choices of κ_i^{ℓ} specify their own basis expansion spaces, and the linear functions can be regarded as a special case associated with the linear kernel $\kappa_i^{\ell}(y, \psi) = y\psi$. An alternative popular kernel is the Gaussian one that is given by $\kappa_i^{\ell}(y, \psi) := \exp[-(y - \psi)^2/(2\sigma^2)]$. Defining the inner product as $\langle \kappa_i^{\ell}(y, \psi_1), \kappa_i^{\ell}(y, \psi_2) \rangle := \sum_{\tau} \kappa_i^{\ell}(y_{\tau}, \psi_1) \kappa_i^{\ell}(y_{\tau}, \psi_2)$, a kernel is reproducing if it satisfies $\langle \kappa_i^{\ell}(y, \psi_1), \kappa_i^{\ell}(y, \psi_2) \rangle = \kappa_i^{\ell}(\psi_1, \psi_2)$, which induces the RKHS norm $\|f_{ij}^{\ell}\|_{\mathcal{H}_i^{\ell}}^2 = \sum_{\tau} \sum_{\tau'} \beta_{ij\tau}^{\ell} \beta_{ij\tau'}^{\ell} \kappa_i^{\ell}(y_{i\tau}, y_{i\tau'})$ [70].

Considering the measurements per node j, with functions $f_{ij}^{\ell} \in \mathcal{H}_i^{l}$, for $i = 1, \ldots, N$ and $\ell = 0, 1, \ldots, L$, the present chapter advocates the following regularized least-squares (LS) estimates of the aforementioned functions obtained as

$$\begin{aligned} \{\hat{f}_{ij}^{\ell}\} = \arg \min_{\{f_{ij}^{\ell} \in \mathcal{H}_i^{\ell}\}} \frac{1}{2} \sum_{t=1}^{T} \Bigg[& y_{jt} - \sum_{i \neq j} a_{ij}^0 f_{ij}^0(y_{it}) \\ & - \sum_{i=1}^{N} \sum_{\ell=1}^{L} a_{ij}^{\ell} f_{ij}^{\ell}(y_{it}) \Bigg]^2 + \lambda \sum_{i=1}^{N} \sum_{\ell=0}^{L} \Omega(\|a_{ij}^{\ell} f_{ij}^{\ell}\|_{\mathcal{H}^{\ell}}) \end{aligned} \tag{5.6}$$

where $\Omega(.)$ denotes a regularizing function, which will be specified later. An important result that will be used in the following is the representer theorem [47, p. 169], according to which the optimal solution for each f_{ij}^{ℓ} in (5.6) is given by

$$\hat{f}_{ij}^{\ell}(y) = \sum_{t=1}^{T} \beta_{ijt}^{\ell} \kappa_i^{\ell}(y, y_{i(t-\ell)}). \tag{5.7}$$

Although the function spaces in (5.5) include infinite basis expansions, since the given data are finite, namely T per node, the optimal solution in (5.7) entails a finite basis expansion. Substituting (5.7) into (5.6), and letting $\boldsymbol{\beta}_{ij}^{\ell} := [\beta_{ij1}^{\ell}, \ldots, \beta_{ijT}^{\ell}]^{\top}$, and $\boldsymbol{\alpha}_{ij}^{\ell} := a_{ij}^{\ell} \boldsymbol{\beta}_{ij}^{\ell}$, the functional minimization in (5.6) boils down to optimizing over vectors $\{\boldsymbol{\alpha}_{ij}^{\ell}\}$. Specifically, (5.6) can be equivalently written in vector form as

$$\{\hat{\boldsymbol{\alpha}}_{ij}^{\ell}\} = \arg\min_{\{\boldsymbol{\alpha}_{ij}^{\ell}\}} \frac{1}{2}\left\|\mathbf{y}_j - \sum_{i\neq j}\mathbf{K}_i^0\boldsymbol{\alpha}_{ij}^0 - \sum_{i=1}^{N}\sum_{\ell=1}^{L}\mathbf{K}_i^{\ell}\boldsymbol{\alpha}_{ij}^{\ell}\right\|_2^2 \\ + \lambda\sum_{i=1}^{N}\sum_{\ell=0}^{L}\Omega\left(\sqrt{(\boldsymbol{\alpha}_{ij}^{\ell})^{\top}\mathbf{K}_i^{\ell}\boldsymbol{\alpha}_{ij}^{\ell}}\right) \tag{5.8}$$

where $\mathbf{y}_j := [y_{j1}, \ldots, y_{jT}]^{\top}$, and the $T\times T$ matrices $\{\mathbf{K}_i^{\ell}\}$ have entries $[\mathbf{K}_i^{\ell}]_{t,\tau} = \kappa_i^{\ell}(y_{it}, y_{i(\tau-\ell)})$. Furthermore, collecting all the observations at different nodes in $\mathbf{Y} := [\mathbf{y}_1, \ldots, \mathbf{y}_N] \in \mathbb{R}^{T\times N}$ and letting $\bar{\mathbf{K}}^{\ell} := [\mathbf{K}_1^{\ell} \ldots \mathbf{K}_N^{\ell}]$, (5.8) can be written as

$$\{\boldsymbol{\alpha}_{ij}^{\ell}\} = \arg\min_{\boldsymbol{\alpha}_{ii}^0=\mathbf{0},\{\boldsymbol{\alpha}_{ij}^{\ell}\}} \frac{1}{2}\left\|\mathbf{Y} - \sum_{l=1}^{L}\bar{\mathbf{K}}^{\ell}\mathbf{W}_{\alpha}^{\ell}\right\|_F^2 + \lambda\sum_{i=1}^{N}\sum_{\ell=0}^{L}\Omega\left(\sqrt{(\boldsymbol{\alpha}_{ij}^{\ell})^{\top}\mathbf{K}_i^{\ell}\boldsymbol{\alpha}_{ij}^{\ell}}\right) \tag{5.9}$$

where the $NT \times N$ block matrix

$$\mathbf{W}_{\alpha}^{\ell} := \begin{bmatrix} \boldsymbol{\alpha}_{11}^{\ell} & \cdots & \boldsymbol{\alpha}_{1N}^{\ell} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\alpha}_{N1}^{\ell} & \cdots & \boldsymbol{\alpha}_{NN}^{\ell} \end{bmatrix} \tag{5.10}$$

exhibits a structure 'modulated' by the entries of $\mathbf{A}^{\ell}$. For instance, if $a_{ij}^{\ell} = 0$, then $\boldsymbol{\alpha}_{ij}^{\ell} := a_{ij}^{\ell}\boldsymbol{\beta}_{ij}^{\ell}$ is an all-zero block, irrespective of the values taken by $\boldsymbol{\beta}_{ij}^{\ell}$.

Instead of the LS cost used in (5.6) and (5.9), alternative loss functions could be employed to promote robustness using the ϵ-insensitive, or, the ℓ_1-error norm; see e.g., [47, Ch. 12]. Regarding the regularizing function $\Omega(.)$, typical choices are $\Omega(z) = |z|$, or, $\Omega(z) = z^2$. The former is known to promote sparsity of edges, which is prevalent to most networks; see e.g., [107]. In principle, leveraging such prior knowledge naturally leads to more efficient topology estimators, since $\{\mathbf{A}^{\ell}\}$ are promoted to have only a few nonzero entries. The sparse nature of $\mathbf{A}^{\ell}$ manifests itself as *block sparsity* in $\mathbf{W}_{\alpha}^{\ell}$. Specifically, using $\Omega(z) = |z|$, one obtains the following estimator of the coefficient vectors $\{\boldsymbol{\alpha}_{ij}^{\ell}\}$ for $j = 1, \ldots, N$

$$\{\hat{\boldsymbol{\alpha}}_{ij}^{\ell}\} = \arg\min_{\hat{\boldsymbol{\alpha}}_{ii}^0=\mathbf{0},\{\boldsymbol{\alpha}_{ij}^{\ell}\}} \frac{1}{2}\left\|\mathbf{Y} - \sum_{l=1}^{L}\bar{\mathbf{K}}^{\ell}\mathbf{W}_{\alpha}^{\ell}\right\|_F^2 + \lambda\sum_{\ell=0}^{L}\sum_{j=1}^{N}\sum_{i=1}^{N}\sqrt{(\boldsymbol{\alpha}_{ij}^{\ell})^{\top}\mathbf{K}_i^{\ell}\boldsymbol{\alpha}_{ij}^{\ell}}. \tag{5.11}$$

Recognizing that summands in the regularization term of (5.11) can be written as $\sqrt{(\boldsymbol{\alpha}_{ij}^{\ell})^{\top}\mathbf{K}_i^{\ell}\boldsymbol{\alpha}_{ij}^{\ell}} = \|(\mathbf{K}_i^{\ell})^{1/2}\boldsymbol{\alpha}_{ij}^{\ell}\|_2$, which is the weighted ℓ_2-norm of $\boldsymbol{\alpha}_{i,j}$, the entire regularizer can henceforth be regarded as the weighted $\ell_{2,1}$-norm of $\mathbf{W}_{\alpha}^{\ell}$, that is known to be useful for promoting block sparsity. It is clear that (5.11) is a convex problem, which admits a globally optimal solution. In fact, the problem structure of (5.11) lends itself naturally to efficient iterative *proximal optimization* methods e.g., proximal gradient descent iterations [170, Ch. 7], or, the alternating direction method of multipliers (ADMM) [171].

For a more detailed description of algorithmic approaches adopted to unveil the hidden topology by solving (5.11), the reader is referred to Appendix 5.8.1. All in all, Algorithm 6b is a summary of the novel iterative solver of (5.11) derived based on ADMM iterations. Per iteration, the complexity of ADMM is in the order of $\mathcal{O}(T^2NL)$, which is linear in the network size N. A couple of remarks are now in order.

Remark 1: Selecting $\Omega(z) = z^2$ is known to control model complexity, and thus prevent overfitting [47, Ch. 3]. Let $\mathbf{D}^{\ell}$:=Bdiag$(\mathbf{K}_1^{\ell}\ldots\mathbf{K}_N^{\ell})$, and $\mathbf{D}$:=Bdiag$(\mathbf{D}^0\ldots\mathbf{D}^L)$, where Bdiag(.) is a block diagonal of its matrix arguments. Substituting $\Omega(z) = z^2$ into (5.9), one obtains

$$\{\hat{\boldsymbol{\alpha}}_{ij}^{\ell}\} = \underset{\hat{\boldsymbol{\alpha}}_{ii}^0=\mathbf{0},\{\boldsymbol{\alpha}_{ij}^{\ell}\}}{\arg\min} \quad \frac{1}{2}\left\|\mathbf{Y} - \bar{\mathbf{K}}\mathbf{W}_{\alpha}\right\|_F^2 + \lambda \ \text{trace}(\mathbf{W}_{\alpha}^{\top}\mathbf{D}\mathbf{W}_{\alpha}) \tag{5.12}$$

where $\bar{\mathbf{K}} := [\bar{\mathbf{K}}^0\ldots\bar{\mathbf{K}}^L]$, and $\mathbf{W}_{\alpha} := [(\mathbf{W}_{\alpha}^0)^{\top}\ldots(\mathbf{W}_{\alpha}^L)^{\top}]^{\top}$. Problem (5.12) is convex and can be solved in closed form as

$$\hat{\bar{\boldsymbol{\alpha}}}_j = \left(\bar{\mathbf{K}}_j^{\top}\bar{\mathbf{K}}_j + 2\mathbf{D}_j\right)^{-1}\bar{\mathbf{K}}_j^{\top}\mathbf{y}_j \tag{5.13}$$

where $\bar{\boldsymbol{\alpha}}_j$ denotes the $(NL-1)T \times 1$ vector obtained after removing entries of the j-th column of $\mathbf{W}_{\alpha}$ indexed by $\mathcal{I}_j := \{(j-1)T+1,\ldots,jT\}$; $\bar{\mathbf{K}}_j$ collects columns of $\bar{\mathbf{K}}$ excluding the columns indexed by $\mathcal{I}_j$; and the block-diagonal matrix $\mathbf{D}_j$ is obtained after eliminating rows and columns of $\mathbf{D}$ indexed by $\mathcal{I}_j$. Using the matrix inversion lemma, the complexity of solving (5.13) is in the order of $\mathcal{O}(T^3NL)$.

Remark 2: Relying on an operator kernel (OK), the approach in [164] offers a more general nonlinear VARM (but not SVARM) than the one adopted here. However, [164] did not account for instantaneous or the multiple-lagged effects. Meanwhile, estimating $\bar{\mathbf{f}}(\mathbf{y}_{t-1})$ in [164] does not scale well as the size of the network (N) increases. Also OK-VARM is approximated in [164] using the Jacobian, which again adds to the complexity of the algorithm, and may degrade the generality of the proposed model. Finally, the model in [164] is limited in its ability to incorporate the structure of the network (e.g., edge sparsity). In order to incorporate prior information on the model structure, [164] ends up solving a nonconvex problem, which might experience local minima, and the flexibility in choosing kernel functions will also be sacrificed. In contrast, our approach entails a natural extension to a data-driven kernel selection, which will be outlined in the next section.

Remark 3: Estimation of a nonlinear function generally requires a large number of samples (T), consequently incurring increased complexity. Clearly, the proposed method is prone to scalability issues when dealing with even moderately-sized networks. This motivates solving the kernel-based optimization problem "on the fly,"; see also [15,84] for recent examples. However, deriving such real-time solvers is beyond the scope of the present chapter, whose focus is the novel nonlinear modeling framework. Only batch algorithms have been presented, but pursuit of more efficient online algorithms constitutes an important future direction.

5.5 Data-driven kernel selection

Choice of the kernel function determines the associated Hilbert space, and it is therefore of significant importance in estimating the nonlinear functions $\{f_{ij}^\ell\}$. Although Section 5.4 assumed that the kernels $\{\kappa_i^\ell\}$ are available, this is not the case in general, and this section advocates a data-driven strategy for selecting them. Given a dictionary of reproducing kernels $\{\kappa_p\}_{p=1}^P$, it has been shown that any function in the convex hull $\mathcal{K} := \{\kappa | \kappa = \sum_{p=1}^P \theta_p \kappa_p,\ \theta_p \geq 0,\ \sum_{p=1}^P \theta_p = 1\}$ is a reproducing kernel [90]. Therefore, the goal of the present section is to select a kernel from $\mathcal{K}$ that best fits the data. For ease of exposition, consider $\kappa_i^\ell = \kappa \in \mathcal{K}$, for all $\ell = 0, 1, \ldots, L$ and $i = 1, \ldots, N$ in (5.6), therefore $\mathcal{H}_i^\ell = \mathcal{H}^{(\kappa)}$. Note that the formulation can be readily extended to settings when $\{\kappa_i^\ell\}$ are different. Incorporating κ as a variable function in (5.6) yields

$$\{\hat{f}_{ij}^\ell\} = \arg \min_{\kappa \in \mathcal{K}, \{f_{ij}^\ell \in \mathcal{H}^{(\kappa)}\}} \frac{1}{2} \sum_{t=1}^{T} \Bigg[y_{jt} - \sum_{i \neq j} a_{ij}^0 f_{ij}^0(y_{it}) - \sum_{i=1}^{N} \sum_{\ell=1}^{L} a_{ij}^\ell f_{ij}^\ell(y_{it}) \Bigg]^2 + \lambda \sum_{i=1}^{N} \sum_{\ell=0}^{L} \Omega(\|a_{ij}^\ell f_{ij}^\ell\|_{\mathcal{H}^{(\kappa)}}) \tag{5.14}$$

Algorithm 6b ADMM for network topology identification

```
1: Input: $\mathbf{Y}$, $\{\{\mathbf{K}_i^\ell\}_{i=1}^N\}_{\ell=1}^L$, $\tau_\alpha$, $\lambda$, $\rho$
2: Initialize: $\boldsymbol{\Gamma}[0] = \mathbf{0}_{NT\times N}$, $\boldsymbol{\Xi}[0] = \mathbf{0}_{NT\times N}$, $k = 0$
3: for $\ell = 1, \ldots, L$ do
4:     $\mathbf{D}^\ell$=Bdiag$(\mathbf{K}_1^\ell, \ldots, \mathbf{K}_N^\ell)$
5:     $\bar{\mathbf{K}}^\ell = [\mathbf{K}_1^\ell \ldots \mathbf{K}_N^\ell]$
6: end for
7: $\bar{\mathbf{K}} := [\bar{\mathbf{K}}^0 \ldots \bar{\mathbf{K}}^L]$, $\mathbf{D}$=Bdiag$(\mathbf{D}^0 \ldots \mathbf{D}^L)$
8: for $j = 1, \ldots, N$ do
9:     $\mathcal{I}_j := \{(j-1)T + 1, \ldots, jT\}$
10:    $\bar{\mathcal{I}}_j := \{(k,l) | k \notin \mathcal{I}_j$, or $l \notin \mathcal{I}_j\}$
11:    $\tilde{\mathcal{I}}_j := \{(k,l) | l \notin \mathcal{I}_j\}$
12:    $\mathbf{D}_j = [\mathbf{D}]_{\bar{\mathcal{I}}_j}$, $\bar{\mathbf{K}}_j = [\bar{\mathbf{K}}]_{\tilde{\mathcal{I}}_j}$
13: end for
14: while not converged do
15:    for $j = 1, \ldots, N$ (in parallel) do
16:        $\mathbf{q}_j[k] = \rho\mathbf{D}_j^{1/2}\boldsymbol{\gamma}_j[k] + \bar{\mathbf{K}}_j^\top\mathbf{y}_j - \mathbf{D}_j^{1/2}\boldsymbol{\xi}_j[k]$
17:        $\bar{\boldsymbol{\alpha}}_j[k+1] = \left(\bar{\mathbf{K}}_j^\top\bar{\mathbf{K}}_j + \rho\mathbf{D}_j\right)^{-1}\mathbf{q}_j[k]$
18:        $\boldsymbol{\gamma}_{ij}^\ell[k] = \mathcal{P}_{\lambda/\rho}\left((\mathbf{K}_i^\ell)^{1/2}\boldsymbol{\alpha}_{ij}^\ell[k+1] + \boldsymbol{\xi}_{ij}^\ell[k]/\rho\right)$,
19:         for $i = 1, \ldots N$, $\ell = 0, \ldots L$
20:    end for
21:    $\mathbf{W}_\alpha[k+1] := [(\mathbf{W}_\alpha^0)^\top[k+1], \ldots, (\mathbf{W}_\alpha^L)^\top[k+1]]^\top$
22:    $\boldsymbol{\Gamma}[k+1] := [(\boldsymbol{\Gamma}^0[k+1])^\top, \ldots, (\boldsymbol{\Gamma}^L[k+1])^\top]^\top$
23:    $\boldsymbol{\Xi}[k+1] = \boldsymbol{\Xi}[k] + \rho(\mathbf{D}^{1/2}\mathbf{W}_\alpha[k+1] - \boldsymbol{\Gamma}[k+1])$
24:    $k = k + 1$
25: end while
26: Edge identification:(after converging to $\hat{\boldsymbol{\alpha}}_{ij}^*$)
27:  $\hat{a}_{ij}^* \neq 0$ if $\|\hat{\boldsymbol{\alpha}}_{ij}^*\| \geq \tau_\alpha$, else $\hat{a}_{ij}^* = 0$, $\forall\,(i,j)$
28: return $\{\hat{\mathbf{A}}^\ell\}_{\ell=0}^L$
```

where $\mathcal{H}^{(\kappa)}$ denotes the Hilbert space associated with kernel function κ. With $\mathcal{H}_p$ denoting the RKHS induced by κ_p, it has been shown in [65] and [90] that the optimal $\{\hat{f}_{ij}^\ell\}$ in (5.14) is expressible in a separable form as

$$\hat{f}_{ij}^\ell(y) := \sum_{p=1}^P f_{ij}^{\ell,p}(y) \tag{5.15}$$

where $f_{ij}^{\ell,p}$ belongs to RKHS $\mathcal{H}_p$, for $p = 1, \ldots, P$. Substituting (5.15) into (5.14), one obtains

$$\{\hat{f}_{ij}^{\ell}\} = \arg \min_{\{f_{ij}^{\ell,p} \in \mathcal{H}_p\}} \frac{1}{2} \sum_{t=1}^{T} \Bigg[y_{jt} - \sum_{i \neq j} \sum_{p=1}^{P} a_{ij}^{0} f_{ij}^{0,p}(y_{it}) \tag{5.16}$$

$$- \sum_{i=1}^{N} \sum_{\ell=1}^{L} \sum_{p=1}^{P} a_{ij}^{\ell} f_{ij}^{\ell,p}(y_{it}) \Bigg]^2 + \lambda \sum_{i=1}^{N} \sum_{\ell=0}^{L} \sum_{p=1}^{P} \Omega(\|a_{ij}^{\ell} f_{ij}^{\ell,p}\|_{\mathcal{H}_p}).$$

Note that (5.16) and (5.6) have similar structure, and their only difference pertains to an extra summation over P candidate kernels. Hence, (5.16) can be solved in an efficient manner along the lines of the iterative solver of (5.6) listed under Algorithm 6b [cf. the discussion in Section 5.4]. Further details of the solution are omitted due to space limitations.

Remark 4: Note that $\{\theta_p\}$ does not show up in the optimization problem in (5.16), since all the coefficients can be readily absorbed into the nonlinear functions without affecting optimality. This is a consequence of the property that given any function belonging to a prescribed RKHS, all its scaled versions belong to the same RKHS [c.f. (5.5)]; see also [90] for a detailed proof.

5.6 Numerical tests

This section presents results from numerical tests conducted on both synthetic and real data to corroborate the effectiveness of the proposed approach. Simulated data were generated via a different model in order to assess the impact of the presence of nonlinear dependencies. Tests on real data were based on seizure experiments captured from a number of subjects.

5.6.1 Synthetic data tests

Data generation. Setting $L = 1$, synthetic data were generated via a random 20-node ($N = 20$) Erdős-Rényi graph, with edge probability 0.4. The resulting graph was encoded as a binary 20×20 adjacency matrix. Using this graph, simulated data were generated via both linear and nonlinear models. For several values of T, entries of $\mathbf{Y} \in \mathbb{R}^{N \times T}$ were randomly sampled from the standardized normal distribution, that is, $y_{nt} \sim \mathcal{N}(0, 1)$. Furthermore, matrices $\{\mathbf{K}_m^{\ell}\}$ were generated using prescribed kernels, that is, entry (i, j) of $\mathbf{K}_m$ was set to $[\mathbf{K}_t^{\ell}]_{ij} = k(y_{it}, y_{jt})$, where the kernel function $k(\cdot, \cdot)$ is known a priori. Entries of coefficient vectors $\boldsymbol{\alpha}_{ij} \in \mathbb{R}^T$ were drawn independently from $\mathcal{N}(0, 1)$, while noise terms were generated i.i.d. as $e_{ij} \sim \mathcal{N}(0, \sigma_e^2)$.

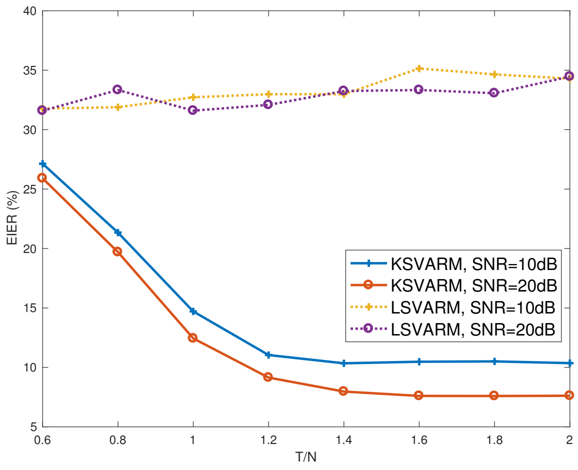


Figure 5.2: Plot of EIER vs. measurement ratio (T/N), with simulated data generated via a polynomial kernel of order $P = 2$. Note that K-SVARMs consistently outperform LSVARMs.

Experiments were run for different values of T, with the edge detection threshold τ selected in each setting to obtain the lowest edge identification error rate (EIER), defined as

$$\text{EIER} := \frac{\|\mathbf{A} - \hat{\mathbf{A}}\|_0}{N(N-1)} \times 100\%. \tag{5.17}$$

The operator $\|\cdot\|_0$ denotes the number of nonzero entries of its argument. For all experiments, error plots were generated with values of EIER averaged over 100 independent runs.

Test results. Figures 5.2 and 5.3 depict EIER values plotted against the measurement ratio (T/N) under varying signal-to-noise ratios (SNR), for polynomial and Gaussian kernels, respectively. The synthetic graph was generated with edge probability $p = 0.3$. Figure 5.2 plots the EIER when data are generated by (5.4), using a polynomial kernel of order $P = 2$, and Figure 5.3 plots the error performance realized with data generated via a Gaussian kernel with bandwidth $\sigma^2 = 1$. It is clear that adoption of nonlinear SVARMs yields markedly better performance than topology inference approaches based on linear SVARMs, which corroborates the effectiveness of the proposed algorithm in identifying the network topology when the dependencies among nodes are nonlinear. It is also worth observing that as SNR decreases, the performance of the proposed algorithm deteriorates slightly, but can still yield much better performance than the linear approach.

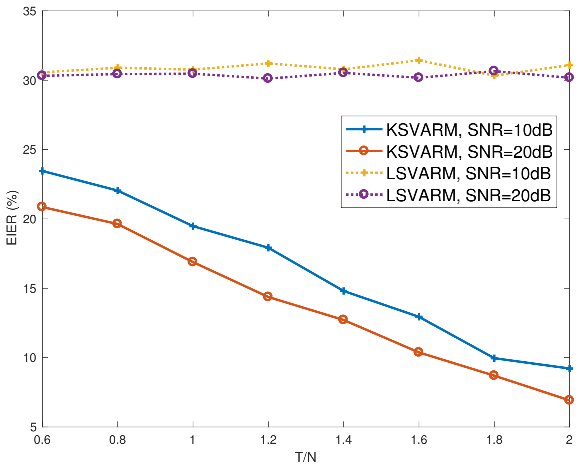


Figure 5.3: Plot of EIER vs. (T/N) with data generated using a Gaussian kernel with $\sigma^2 = 1$; K-SVARMs uniformly lead to lower errors than linear LSVARMs over varying SNR levels, based on empirical observations.

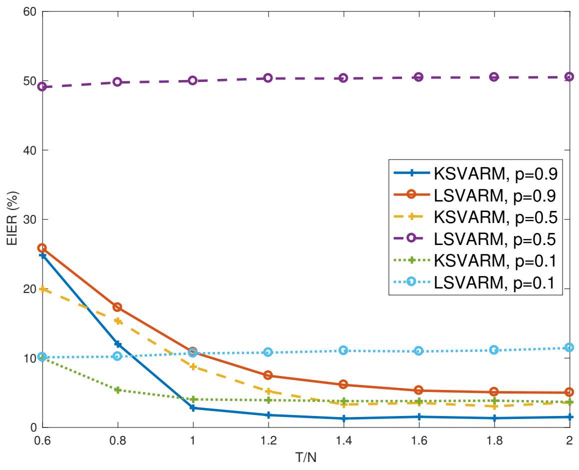
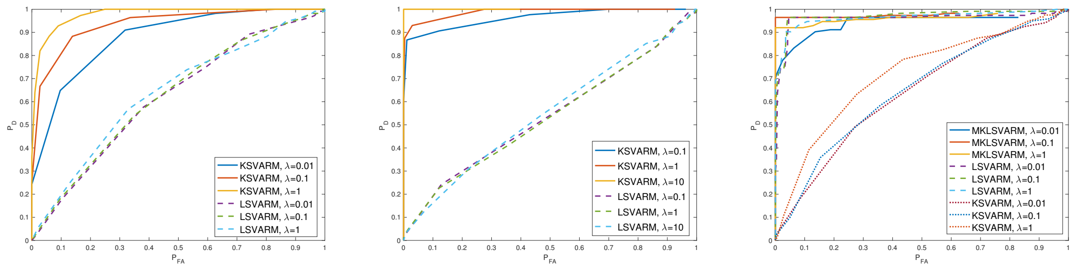


Figure 5.4: Plot of EIER vs. (T/N) with simulated data generated via a polynomial kernel of order $P = 2$; it can be empirically observed that K-SVARMs uniformly outperform LSVARMs across varying edge densities.



(a) (b) (c)

Figure 5.5: ROC curves generated under different modeling assumptions: a) K-SVARM based on a Gaussian kernel with $\sigma^2 = 1$; b) K-SVARM based on polynomial kernel of order $P = 2$; and c) Linear SVARM.

In order to assess edge detection performance, receiver operating characteristic (ROC) curves are plotted under different modeling assumptions in Figure 5.5. With P_D denoting the probability of detection, and P_{FA} the probability of false alarms, each point on the ROC corresponds to a pair (P_{FA}, P_D) for a prescribed threshold. Figure 5.5 (a) results from tests run on data generated by Gaussian kernels with $\sigma^2 = 1$, while Figure 5.5 (b) corresponds to polynomial kernels of order $P = 2$. Using the *area under the curve* (AUC) as the edge-detection performance criterion, Figures 5.5 (a) and (b) clearly emphasize the benefits of accounting for nonlinearities. In both plots, kernel-based approaches result in the higher AUC metrics than approaches resorting to linear SVARMs.

Figure 5.5 (c) plots ROC curves based on linear and kernel based SVARMs, with simulated data actually generated using a linear SVARM. The curves are parameterized by the sparsity-control parameter λ. Not surprisingly, kernel-based SVARMs adopting polynomial kernels underperform the linear SVARM, due to the inherent model mismatch. However, the kernel SVARM endowed with a multi-kernel learning scheme (MK-SVARM) is shown to attain comparable performance to the linear SVARM when the prescribed dictionary comprises both linear and polynomial kernels.

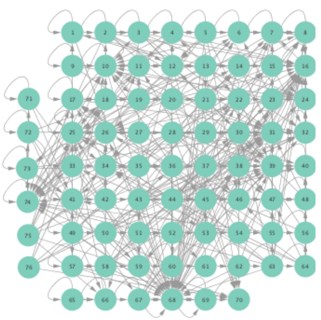
5.6.2 Real data tests

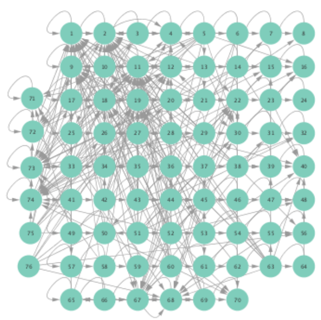
This section presents test results on seizure data, captured through experiments conducted in an *epilepsy* study [172]. Epilepsy refers to a chronic neurological condition characterized by recurrent seizures, globally afflicting over 20 million people, and often associated with abnormal neuronal activity within the brain. Diagnosis of the condition sometimes involves comparing EEG or ECoG time series obtained from a patient's brain before and after onset of a seizure. Recent studies have shown increasing interest in analysis of connectivity networks inferred from the neuronal time series, in order to gain more insights about the unknown physiological mechanisms underlying epileptic seizures. In this section, connectivity networks are inferred from the seizure data using the novel approach, and a number of comparative measures are computed from the identified network topologies.

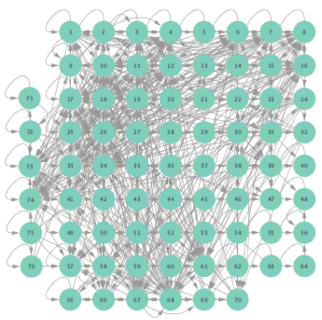
Seizure data description. Seizure data were obtained for a 39-year-old female subject with a case of intractable epilepsy at the *University of California, San Francisco (UCSF) Epilepsy Center*; see also [172]. An 8×8 subdural electrode grid was implanted into the cortical surface of the subject's brain, and two accompanying electrode strips, each comprising six electrodes (a.k.a., depth electrodes) were implanted deeper into the brain. Over a period of five days, the combined electrode network recorded 76 ECoG time series, consisting of voltage levels measured in a region within close proximity of each electrode.

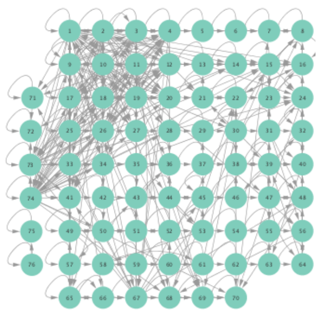
ECoG epochs containing eight seizures were extracted from the record and analyzed by a specialist. The time series at each electrode were first passed through a bandpass filter, with cut-off frequencies of 1 and 50 Hz, and the so-termed *ictal* onset of each seizure was identified as follows. A board-certified neurophysiologist identified the initial manifestation of rhythmic high-frequency, low-voltage focal activity, which characterizes the onset of a seizure. Samples of data before and after this seizure onset were then extracted from the ECoG time series. The per-electrode time series were then divided into 1s windows, with 0.5s overlaps between consecutive windows, and the average spectral power between 5Hz and 15Hz was computed per window. Finally, power spectra over all electrodes were averaged, and the ictal onset was identified by visual inspection of a dramatic increase (by at least an order of magnitude) in the average power. Two temporal intervals of interest were picked for further analysis, namely, the *preictal* and ictal intervals. The preictal interval is defined as a 10s interval preceding seizure onset, while the ictal interval comprises the 10s immediately afterwards. Further details about data acquisition and pre-processing are provided in [172].

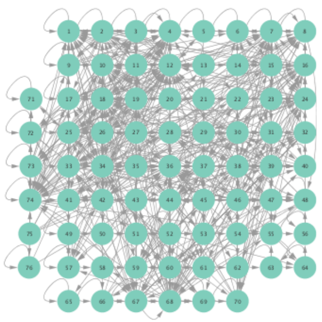
The goal here was to assess whether modeling nonlinearities, and adopting the novel kernel-based approach would yield significant insights pertaining to causal/effective dependencies between brain regions, that linear variants would otherwise fail to capture. Toward this goal, several standard network analysis measures were adopted to characterize the structural properties


(a)


(b)


(c)


(d)


(e)

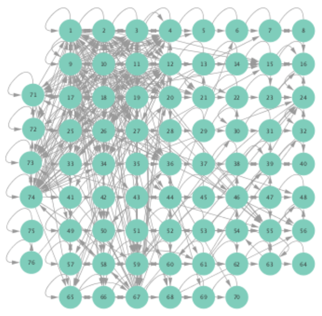

(f)

Figure 5.6: Visualizations of networks inferred from ECoG data: (a) linear SVARM with $L = 1$ on preictal time series; (b) linear SVARM on ictal time series; (c) K-SVARM on preictal time series, using Gaussian kernel with $\sigma = 1$; (d) the same K-SVARM on ictal time series; (e) K-SVARM with kernel selection on preictal time series; and finally (f) K-SVARM with kernel selection on ictal time series.

of the inferred networks.

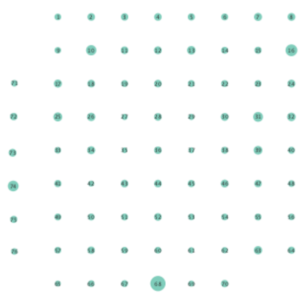
Inferred networks. Prior to running the developed algorithm, 10s intervals were chosen from the preprocessed ECoG data with the sampling rate set to 400Hz, and then divided into 20 successive segments, each comprising 200 data samples over a 0.5s horizon. To illustrate this, suppose the 10s interval starts from $t = 0$s and ends at $t = 10s$, then the first segment comprises samples taken over the interval $[0s, 0.5s]$, the second one would be $[0.5s, 1s]$, and so on. After this segmentation of the time series, directed network topologies were inferred using Algorithm 6b with $L = 1$, based on the 0.5s segments, instead of the entire signal, to ensure that the signal is approximately *stationary* per experiment run. A directed link from electrode i to j was drawn if at least one of the estimates of a_{ij}^{ℓ} turned out to be nonzero.

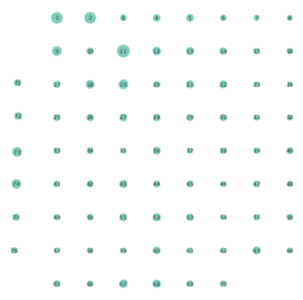
Upon inference of the 20 networks the data pertaining to each seizure, presence/absence of an edge is established via a *t-test* with $P_{FA} = 0.1$. Inference results were not averaged since different seizures may originate from disparate parts of the brain, leading to non-trivial differences between connectivity patterns among electrodes.

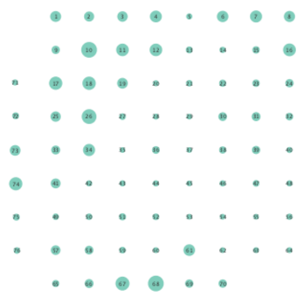
Networks inferred from the preictal and ictal intervals were compared using linear, the kernel-based (K-)SVARMs, and K-SVARM with data-driven kernel selection. The lag lengths were set to $L = 1$ for all cases. For K-SVARM, a Gaussian kernel with $\sigma = 1$ was selected, and with $\rho = 10$, Algorithm 6b with regularization parameter λ was selected via cross-validation. For the data-driven kernel selection scheme, two candidate kernels were employed, namely, a linear kernel, and a polynomial kernel of order 2.

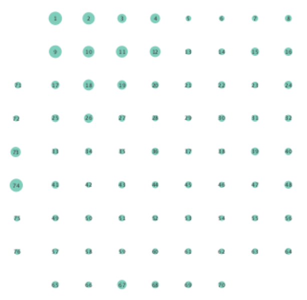
Figure 5.6 depicts networks inferred from different algorithms for both preictal and ictal intervals of the time series. The figure illustrates results obtained by the linear SVARM, and the K-SVARM approach with and without kernel selection. Each node in the network is representative of an electrode, and it is depicted as a circle, while the node arrangement is forced to remain consistent across the six visual representations. A cursory inspection of the visual maps reveals significant variations in connectivity patterns between ictal and preictal intervals for both models. Specifically, networks inferred via the K-SVARMs, reveal a global decrease in the number of links emanating from each node, while those inferred via the linear model depict increases and decreases in links connected to different nodes. Interestingly, the K-SVARM with kernel selection recovered most of the edges inferred by the linear and the K-SVARM using a Gaussian kernel, which implies that both linear and nonlinear interactions may exist in brain networks. Moreover, network topologies inferred via K-SVARM with kernel selection are more similar to those obtained using a single kernel than a linear SVARM, implying that the simple linear model is insufficient to capture the network topology in complex brain networks. However, one is unlikely to gain much insight only by visual inspection of the network topologies. Moreover, note that the To further analyze differences between inferred networks from both models, and to assess the potential benefits gained by adopting the novel scheme, several network topology metrics are computed and compared in the next subsection.

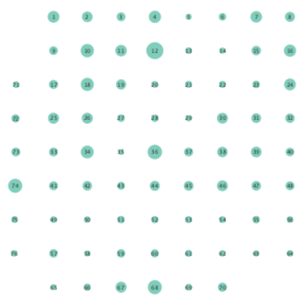
Comparison of network metrics. First, in- and out-degree was computed for nodes in each of the inferred networks. Note that the in-degree of a node counts its number of incoming edges, while the out-degree counts the number of out-going edges. The total degree per node sums the in- and out-degrees, and is indicative of how well-connected a given node is. Figure 5.7 depicts nodes in the network and their total degrees encoded by the radii of circles associated with the


(a)


(b)


(c)


(d)


(e)

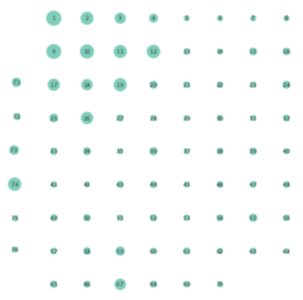

(f)

Figure 5.7: Node degrees of networks inferred from ECoG data encoded by circle radii: (a) linear SVARM on preictal data; (b) linear SVARM on ictal data; (c) K-SVARM on preictal time series; (d) K-SVARM on ictal data; (e) MKL-SVARM on preictal time series; (f) MKL-SVARM on ictal time series.

nodes. As expected from the previous subsection, Figures 5.7 (a) and (b) demonstrate that the linear SVARM yields both increases and deceases in the inferred node degree. On the other hand, the nonlinear SVARM leads to a more spatially consistent observation with most nodes exhibiting a smaller degree after the onset of a seizure (see Figures 5.7 (c) and (d)), which may imply that causal dependencies thin out between regions of the brain once a seizure starts, the same trend is also revealed by the K-SVARM with kernel selection (see Figures 5.7 (e) and (f)).

In order to assess the reachability of brain regions before and after seizure onset, comparisons of the so-termed *average shortest path* lengths were done. Average shortest path of a node computes the average length of shortest paths between the given node and all other nodes; see e.g., [9] for more details. The per-node average shortest path length for each inferred network is depicted in Figure 5.8, with node radii similarly encoding the computed values. Little variation

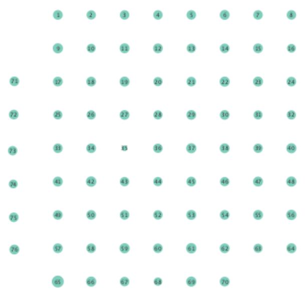
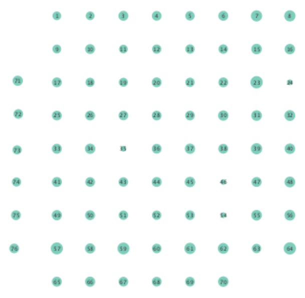
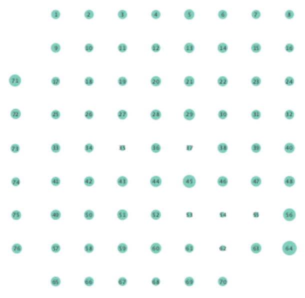
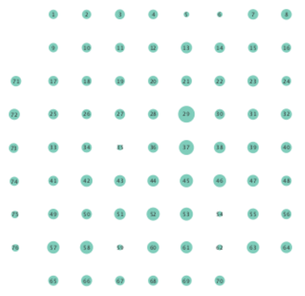
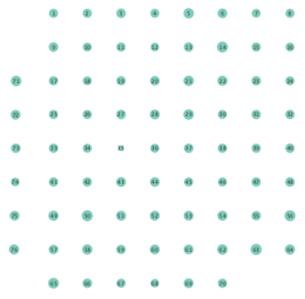
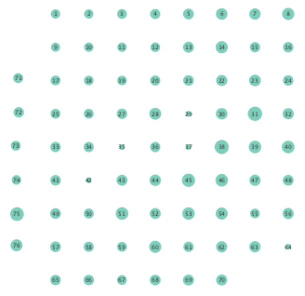

(a)

(b)

(c)

(d)

(e)

(f)

Figure 5.8: Same as in Figure 5.7 for comparison based on average shortest path length of inferred graphs.

between preictal and ictal average shortest path length is seen for the linear model (Figures 5.8 (a) and (b)), while variations are more marked for the K-SVARM, see Figures 5.8 (c-f). It can be seen that modeling nonlinearities reveals subtle changes in reachability of nodes between preictal and ictal phases.

In addition to the local metrics, a number of global measures were computed over entire inferred networks, and pertinent comparisons were drawn between the two phases; see Table 5.1 for a summary of the average global measures of the inferred networks from 8 different seizures. Several global metrics were considered, e.g., network density, global clustering coefficient, network diameter, and average number of neighbors.These metrics are obtained by averaging the metrics of networks inferred from 8 seizures.

Network density refers to the number of actual edges divided by the number of potential edges, while the global clustering coefficient is the fraction of connected triplets that form triangles, adjusted by a factor of three to compensate for double counting. On the other hand,

	Linear SVARM		**K-SVARM**		**MKL-SVARM**	
	Preictal	Ictal	Preictal	Ictal	Preictal	Ictal
Network density	0.103	0.095	0.136	0.089	0.148	0.101
Glob. clustering coeff.	0.246	0.220	0.372	0.356	0.391	0.343
No. of connect. comp.	8	7	2	5	2	6
Network diameter	5	5	7	9	4	7
Avg. no. of neighbors	7.73	6.89	10.23	6.71	11.11	7.55

Table 5.1: Comparison of global metrics associated with networks inferred from ECoG seizure data using the linear, K-SVARM, and K-SVARM with kernel selection scheme. Major differences between the computed metrics indicate that one may gain insights from network topologies inferred via models that capture nonlinear dependencies.

network diameter is the length of the longest geodesic, excluding infinity. Table 5.1 shows that networks inferred via the K-SVARMs and MKL-SVARMs exhibit lower network cohesion after seizure onset, as captured by network density, global clustering coefficient, and average number of neighbors, while the network diameter increases. These changes provide empirical evidence that the brain network becomes less connected, and diffusion of information is inhibited after the onset of an epileptic seizure.

5.7 Summary

This chapter put forth a novel nonlinear SVARM framework that leverages kernels to infer effective connectivity networks in the brain. Postulating a generalized additive model with unknown functions to capture the hidden network structure, a novel regularized LS estimator that promotes sparse solutions was advocated. In order to solve the ensuing convex optimization problem, an efficient algorithm that resorts to ADMM iterations was developed, and a data-driven approach was introduced to select the appropriate kernel. Extensive numerical tests were conducted on ECoG seizure data from a study on epilepsy.

In order to assess the utility of the novel approach, several local and global metrics were adopted and computed on networks inferred before and after the onset of a seizure. By observing changes in network behavior that are revealed by standard metrics before and after seizure onset, it is possible identify key structural differences that may be critical to explain the mysteries of epileptic seizures. With this in mind, the chapter focused on identifying structural differences in the brain network that could not be captured by the simpler linear model. Interestingly, empirical results support adoption of a nonlinear modeling perspective when analyzing differences in effective brain connectivity for epilepsy patients. Specifically, adopting the novel kernel-based approach revealed more significant differences between the preictal and ictal phases of ECoG time series. For instance, it turned out that some regions exhibited fewer dependencies, reduced reachability, and weakened information-routing capabilities after the onset of a seizure. Since the kernel-based model includes the linear SVARM as an instance, the conducted experiments suggest that one may gain more insights by adopting the nonlinear model, a conclusion that may yield informative benefits to studies of epilepsy that leverage network science.

This work paves the way for a number of exciting research directions in analysis of brain networks. Although it has been assumed that inferred networks are static, overwhelming evidence suggests that topologies of brain networks are dynamic, and may change over rather short time horizons. Future studies will extend this work to facilitate tracking of dynamic brain networks. Furthermore, the novel approach will be empirically tested on a wider range of neurological illnesses and disorders, and pertinent comparisons will be done to assess the merits of adopting the advocated nonlinear modeling approach.

5.8 Appendix

5.8.1 Topology Inference via ADMM

Given matrices $\mathbf{Y}$ and $\bar{\mathbf{K}} := [\bar{\mathbf{K}}^0 \ldots \bar{\mathbf{K}}^L]$, this section capitalizes on convexity, and the nature of the additive terms in (5.11) to develop an efficient topology inference algorithm. Proximal optimization approaches have recently been shown useful for convex optimization when the cost function comprises the sum of smooth and nonsmooth terms; see e.g., [173]. Prominent among these approaches is the alternating direction method of multipliers (ADMM), upon which the novel algorithm is based; see e.g., [171] for an early application of ADMM to distributed estimation.

For ease of exposition, let the equality constraints ($\boldsymbol{\alpha}_{jj}^{\ell} = \mathbf{0}$) temporarily remain implicit.

Introducing the change of variables $\boldsymbol{\gamma}_{ij}^{\ell} := (\mathbf{K}_i^{\ell})^{1/2}\boldsymbol{\alpha}_{ij}^{\ell}$, problem (5.11) can be recast as

$$\begin{aligned} &\underset{\{\boldsymbol{\alpha}_{ij}^{\ell}\}}{\arg\min}(1/2)\|\mathbf{Y} - \sum_{\ell=0}^{L}\bar{\mathbf{K}}^{\ell}\mathbf{W}_{\alpha}^{\ell}\|_F^2 + \sum_{\ell=0}^{L} g(\mathbf{\Gamma}^{\ell}) \\ &\text{s.t. } \boldsymbol{\gamma}_{ij}^{\ell} - (\mathbf{K}_i^{\ell})^{1/2}\boldsymbol{\alpha}_{ij}^{\ell} = \mathbf{0} \quad \forall i,j,\ell \end{aligned} \tag{5.18}$$

where $\mathbf{\Gamma}^{\ell} := [\boldsymbol{\gamma}_1^{\ell}\ldots\boldsymbol{\gamma}_N^{\ell}]$, $\boldsymbol{\gamma}_j^{\ell} := [(\boldsymbol{\gamma}_{1j}^{\ell})^{\top}\ldots(\boldsymbol{\gamma}_{Nj}^{\ell})^{\top}]^{\top}$, and $g(\mathbf{\Gamma}^{\ell}) := \lambda\sum_{i=1}^{N}\sum_{j=1}^{N}\|\boldsymbol{\gamma}_{ij}^{\ell}\|_2$ is the nonsmooth regularizer. Let $\mathbf{D}^{\ell}$:=Bdiag$(\mathbf{K}_1^{\ell}\ldots\mathbf{K}_N^{\ell})$, and $\mathbf{D}$:=Bdiag$(\mathbf{D}^0\ldots\mathbf{D}^{\ell})$, where Bdiag(.) is a block diagonal of its matrix arguments. One can then write the augmented Lagrangian of (5.18) as

$$\begin{aligned} \mathcal{L}_{\rho}(\mathbf{W}_{\alpha},\mathbf{\Gamma},\mathbf{\Xi}) = (1/2)\|\mathbf{Y} - \bar{\mathbf{K}}\mathbf{W}_{\alpha}\|_F^2 + g(\mathbf{\Gamma}) \\ + \langle\mathbf{\Xi},\mathbf{D}^{1/2}\mathbf{W}_{\alpha} - \mathbf{\Gamma}\rangle + (\rho/2)\|\mathbf{\Gamma} - \mathbf{D}^{1/2}\mathbf{W}_{\alpha}\|_F^2 \end{aligned} \tag{5.19}$$

where $\mathbf{W}_{\alpha} := [(\mathbf{W}_{\alpha}^0)^{\top}\ldots(\mathbf{W}_{\alpha}^L)^{\top}]^{\top}$, and $\mathbf{\Gamma} := [(\mathbf{\Gamma}^0)^{\top}\ldots(\mathbf{\Gamma}^L)^{\top}]^{\top}$. Note that $\mathbf{\Xi}$ is a matrix of dual variables that collects Lagrange multipliers corresponding to the equality constraints in (5.18), $\langle\mathbf{P},\mathbf{Q}\rangle$ denotes the inner product between $\mathbf{P}$ and $\mathbf{Q}$, while $\rho > 0$ a prescribed penalty parameter. ADMM boils down to a sequence of alternating minimization iterations to minimize $\mathcal{L}_{\rho}(\mathbf{W}_{\alpha},\mathbf{\Gamma},\mathbf{\Xi})$ over the primal variables $\mathbf{W}_{\alpha}$, and $\mathbf{\Gamma}$, followed by a gradient ascent step over the dual variables $\mathbf{\Xi}$; see also [171,174]. Per iteration $k+1$, this entails the following provably-convergent steps, see e.g. [171]

$$\mathbf{W}_{\alpha}[k+1] = \underset{\mathbf{W}_{\alpha}}{\arg\min}\ \mathcal{L}_{\rho}(\mathbf{W}_{\alpha},\mathbf{\Gamma}[k],\mathbf{\Xi}[k]) \tag{5.20a}$$

$$\mathbf{\Gamma}[k+1] = \underset{\mathbf{\Gamma}}{\arg\min}\ \mathcal{L}_{\rho}(\mathbf{W}_{\alpha}[k+1],\mathbf{\Gamma},\mathbf{\Xi}[k]) \tag{5.20b}$$

$$\mathbf{\Xi}[k+1] = \mathbf{\Xi}[k] + \rho(\mathbf{D}^{1/2}\mathbf{W}_{\alpha}[k+1] - \mathbf{\Gamma}[k+1]). \tag{5.20c}$$

Focusing on $\mathbf{W}_{\alpha}[k+1]$, note that (5.20a) decouples across columns of $\mathbf{W}_{\alpha}$, and admits closed-form, parallelizable solutions. Incorporating the structural constraint $\boldsymbol{\alpha}_{jj}^0 = \mathbf{0}$, one obtains the following decoupled subproblem per column j

$$\bar{\boldsymbol{\alpha}}_j[k+1] = \arg\min_{\bar{\boldsymbol{\alpha}}_j}(1/2)\bar{\boldsymbol{\alpha}}_j^{\top}\left(\bar{\mathbf{K}}_j^{\top}\bar{\mathbf{K}}_j + \rho\mathbf{D}_j\right)\bar{\boldsymbol{\alpha}}_j - \bar{\boldsymbol{\alpha}}_j^{\top}\mathbf{q}_j[k] \tag{5.21}$$

where $\mathbf{q}_j[k]$ is constructed by removal of entries indexed by $\mathcal{I}_j$ from $\rho\mathbf{D}^{1/2}\boldsymbol{\gamma}_j[k] + \bar{\mathbf{K}}^\top\mathbf{y}_j - \mathbf{D}^{1/2}\boldsymbol{\xi}_j[k]$, with $\boldsymbol{\xi}_j[k]$ denoting the j-th column of $\boldsymbol{\Xi}[k]$. Assuming $\left(\bar{\mathbf{K}}_j^\top\bar{\mathbf{K}}_j + \rho\mathbf{D}_j\right)$ is invertible, the per-column subproblem (5.21) admits the following closed-form solution per j

$$\bar{\boldsymbol{\alpha}}_j[k+1] = \left(\bar{\mathbf{K}}_j^\top\bar{\mathbf{K}}_j + \rho\mathbf{D}_j\right)^{-1}\mathbf{q}_j[k]. \tag{5.22}$$

On the other hand, (5.20b) can be solved per component vector $\boldsymbol{\gamma}_{ij}^\ell$, and a closed-form solution can be obtained via the so-termed *block shrinkage* operator for each i and j, namely,

$$\boldsymbol{\gamma}_{ij}^\ell[k] = \mathcal{P}_{\lambda/\rho}\left((\mathbf{K}_i^\ell)^{1/2}\boldsymbol{\alpha}_{ij}^\ell[k+1] + \boldsymbol{\xi}_{ij}^\ell[k]/\rho\right) \tag{5.23}$$

where $\mathcal{P}_\lambda(\mathbf{z}) := (\mathbf{z}/\|\mathbf{z}\|_2)\max(\|\mathbf{z}\|_2 - \lambda, 0)$. Upon convergence, $\{a_{ij}^\ell\}$ can be determined by thresholding $\hat{\boldsymbol{\alpha}}_{ij}^\ell$, and declaring an edge present from i to j, if there exists any $\hat{\boldsymbol{\alpha}}_{ij}^\ell \neq \mathbf{0}$, for $\ell = 1, \ldots, L$.

Chapter 6

Online Graph-Adaptive Learning with Scalability and Privacy

6.1 Introduction

In real-world networks, there are often unavailable nodal function values, due to, e.g., privacy issues. Hence, a topic of great practical importance is to interpolate missing nodal values (class, ranking or function), based on the function values at a subset of observed nodes. Interpolation of nodal function values often relies on the assumption of "smoothness" over the graphs, which implies that neighboring nodes will have similar nodal function values. For example, in social networks, people tend to rate e.g., movies similar to their friends, and in financial networks, companies that trade with each other usually belong to the same category. From this point of view, function estimation over graphs based on partial observations has been investigated extensively, [9, 175–179]. Function estimation has been also pursued in the context of semi-supervised learning, e.g., for transductive regression or classification, see e.g., [180–183]. The same task has been studied recently as signal reconstruction over graphs, see e.g., [184–188], where signal values on unobserved nodes can be estimated by properly introducing a graph-aware prior. Kernel-based methods for learning over graphs offer a unifying framework that includes linear and nonlinear function estimators [186, 189, 190]. The nonlinear methods outperform the linear ones but suffer from the curse of dimensionality [70], rendering them less attractive for large-scale networks.

To alleviate this limitation, a scalable kernel-based approach will be introduced in the present chapter, which leverages the random feature approximation to ensure *scalability* while also allowing *real-time* evaluation of the functions over large-scale dynamic networks. In addition, the novel approach incorporates a data-driven scheme for *adaptive* kernel selection.

Adaptive learning over graphs has been also investigated for tracking and learning over possibly dynamic networks, e.g., [190, 191]. Least mean-squares and recursive least-squares adaptive schemes have been developed in [191], without explicitly accounting for evolving network topologies. In contrast, [190] proposed a kernel-based reconstruction scheme to track time-varying signals over time-evolving topologies, but assumed that the kernel function is selected a priori. All these prior works assume that the network size is fixed.

In certain applications however, new nodes may join the network over time. For example, hundreds of new users are joining Facebook or Netflix every day, and new companies are founded in financial networks regularly. Real-time and scalable estimation of the desired functions on these newly-joining nodes is of great importance. While simple schemes such as averaging over one- or multi-hop neighborhoods are scalable to network size by predicting the value on each newly-coming node as a weighted combination of its multi-hop neighborhoods [192], they do not capture global information over the network. In addition, existing rigorous approaches are in general less efficient in accounting for newly-joining nodes, and need to solve the problem over all nodes, every time new nodes join the network, which incurs complexity $\mathcal{O}(N^3)$, where N denotes the network size [186, 189]. As a result, these methods are not amenable to real-time evaluation over newly-joining nodes. To this end, the present chapter develops a scalable *online graph-adaptive* algorithm that can efficiently estimate nodal functions on newly-joining nodes 'on the fly.'

Besides scalability and adaptivity, nodes may have firm *privacy* requirements, and may therefore not be willing to reveal who their neighbors are. However, most graph-based learning methods require knowing the entire connectivity pattern, and thus cannot meet the privacy requirements. The novel random feature based approach on the other hand, only requires an encrypted version of each node's connectivity pattern, which makes it appealing for networks with stringent privacy constraints.

In short, we put forth a novel online multikernel learning (MKL) framework for effectively learning and tracking nonlinear functions over graphs. Our contributions are as follows.

c1) A scalable MKL approach is developed to efficiently estimate the nodal function values both on the observed and un-observed nodes of a graph;

c2) The resultant algorithm is capable of estimating the function value of newly incoming nodes with high accuracy without solving the batch problem over all nodes, making it highly scalable as the network size grows, and suitable for nodal function estimation in dynamic networks;
c3) Unlike most existing methods that rely on nodal feature vectors in order to learn the function, the proposed scheme simply capitalizes on the connectivity pattern of each node, while at the same time, nodal feature vectors can be easily incorporated if available; and,
c4) The proposed algorithm does not require nodes to share connectivity patterns. Instead, a privacy-preserving scheme is developed for estimating the nodal function values based on an encrypted version of the nodal connectivity patterns, hence respecting node privacy.

The rest of this chapter is organized as follows. Preliminaries are in Section 6.2, while Section 6.3 presents an online kernel-based algorithm that allows sequential processing of nodal samples. Section 6.4 develops an online MKL scheme for sequential data-driven kernel selection, which allows graph-adaptive selection of kernel functions to best fit the learning task of interest. Section 6.5 provides performance analysis of the proposed algorithm. Finally, results of corroborating numerical tests on both synthetic and real data are presented in Section 6.6, while concluding remarks along with a discussion of ongoing and future directions are given in Section 6.7.

6.2 Kernel-based learning over graphs

Consider a graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ of N nodes, whose topology is captured by a known adjacency matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$. Let $a_{nn'} \in \mathbb{R}$ denote the (n, n') entry of $\mathbf{A}$, which is nonzero only if an edge is present from node n' to n. A real-valued function (or signal) on a graph is a mapping $f : \mathcal{V} \to \mathbb{R}$, where $\mathcal{V}$ is the set of vertices. The value $f(v) = x_v$ represents an attribute of $v \in \mathcal{V}$, e.g., in the context of brain networks, x_{v_n} could represent the sample of an electroencephalogram (EEG), or functional magnetic resonance imaging (fMRI) measurement at region n. In a social network, x_{v_n} could denote the age, political alignment, or annual income of the nth person. Suppose that a collection of noisy samples $\{y_m = x_{v_{n_m}} + e_m\}_{m=1}^{M}$ is available, where e_m models noise, and $M \leq N$ represents the number of measurements. Given $\{y_m\}_{m=1}^{M}$, and with the graph topology known, the goal is to estimate $f(v)$, and thus reconstruct the graph signal at unobserved vertices. Letting $\mathbf{y} := [y_1, \ldots, y_M]^\top$, the observation vector obeys

$$\mathbf{y} = \boldsymbol{\Psi}\mathbf{x} + \mathbf{e} \tag{6.1}$$

where $\mathbf{x} := [x_{v_1}, \ldots, x_{v_N}]^\top$, $\mathbf{e} := [e_1, \ldots, e_M]^\top$, and $\boldsymbol{\Psi} \in \{0,1\}^{M\times N}$ is a sampling matrix with binary entries $[\boldsymbol{\Psi}]_{m,n_m} = 1$ for $m = 1, \ldots, M$, and 0, elsewhere.

Given $\boldsymbol{\Psi}$, $\mathbf{y}$, and $\mathbf{A}$, the goal is to estimate $\mathbf{x}$ over the entire network. To tackle the under-determined system (6.1), consider function f belonging to a reproducing kernel Hilbert space (RKHS) defined as [186, 189]

$$\mathcal{H} := \{f : f(v) = \sum_{n=1}^{N} \alpha_n \kappa(v, v_n), \alpha_n \in \mathbb{R}\} \tag{6.2}$$

where $\kappa : \mathcal{V} \times \mathcal{V} \to \mathbb{R}$ is a pre-selected kernel function. Hereafter, we will let $n_m = m$ for notational convenience, and without loss of generality (wlog). Given $\mathbf{y}$, the RKHS-based estimate is formed as

$$\hat{f} = \arg\min_{f\in\mathcal{H}} \frac{1}{M}\sum_{m=1}^{M} \mathcal{C}(f(v_m), y_m) + \mu\Omega\left(\|f\|_{\mathcal{H}}^2\right) \tag{6.3}$$

where the cost $\mathcal{C}(\cdot,\cdot)$ can be selected depending on the learning task, e.g., the least-squares (LS) for regression, or the logistic loss for classification; $\|f\|_{\mathcal{H}}^2 := \sum_n \sum_{n'} \alpha_n \alpha_{n'} \kappa(v_n, v_{n'})$ is the RKHS norm; $\Omega(\cdot)$ is an increasing function; and, $\mu > 0$ is a regularization parameter that copes with overfitting. According to the definition of graph RKHS in (6.2), the function estimate can be written as $\hat{f}(v) = \sum_{n=1}^{N} \alpha_n \kappa(v, v_n) := \bar{\boldsymbol{\alpha}}^\top \bar{\mathbf{k}}(v)$, where $\bar{\boldsymbol{\alpha}} := [\alpha_1, \ldots, \alpha_N]^\top \in \mathbb{R}^N$ collects the basis coefficients, and $\bar{\mathbf{k}}(v) := [\kappa(v, v_1), \ldots, \kappa(v, v_N)]^\top$. Substituting into the RKHS norm, we find $\|f\|_{\mathcal{H}}^2 := \sum_n \sum_{n'} \alpha_n \alpha_{n'} \kappa(v_n, v_{n'}) = \bar{\boldsymbol{\alpha}}^\top \bar{\mathbf{K}} \bar{\boldsymbol{\alpha}}$, where the $N \times N$ kernel matrix $\bar{\mathbf{K}}$ has entries $[\bar{\mathbf{K}}]_{n,n'} := \kappa(v_n, v_{n'})$; thus, the functional problem (6.3) boils down to

$$\min_{\bar{\boldsymbol{\alpha}}\in\mathbb{R}^N} \frac{1}{M}\sum_{m=1}^{M} \mathcal{C}(\bar{\boldsymbol{\alpha}}^\top \bar{\mathbf{k}}(v_m), y_m) + \mu\Omega\left(\bar{\boldsymbol{\alpha}}^\top \bar{\mathbf{K}} \bar{\boldsymbol{\alpha}}\right). \tag{6.4}$$

According to the representer theorem, the optimal solution of (6.3) admits the finite-dimensional form given by [186, 189]

$$\hat{f}(v) = \sum_{m=1}^{M} \alpha_m \kappa(v, v_m) := \boldsymbol{\alpha}^\top \mathbf{k}(v). \tag{6.5}$$

where $\boldsymbol{\alpha} := [\alpha_1, \ldots, \alpha_M]^\top \in \mathbb{R}^M$, and $\mathbf{k}(v) := [\kappa(v, v_1), \ldots, \kappa(v, v_M)]^\top$. This means that the coefficients corresponding to the unobserved nodes are all zeros. This implies that the function over the graph can be estimated by optimizing over the $M \times 1$ vector $\boldsymbol{\alpha}$ [cf. (6.3)]

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^M} \frac{1}{M} \sum_{m=1}^{M} \mathcal{C}(\boldsymbol{\alpha}^\top \mathbf{k}(v_m), y_m) + \mu \Omega \left(\boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha} \right) \tag{6.6}$$

where $\mathbf{K} := \boldsymbol{\Psi}^\top \bar{\mathbf{K}} \boldsymbol{\Psi}$. For general kernel-based learning tasks, $\bar{\mathbf{K}}$ is formed using the nonlinear functions of pairwise correlations $\kappa(v_n, v_{n'}) = \boldsymbol{\phi}_n^\top \boldsymbol{\phi}_{n'}$, where $\boldsymbol{\phi}_n$ denotes the feature vector of node n, which can collect, for example, the buying history of users on Amazon, or the trading history of companies in financial networks. However, such information may not be available in practice, due to, e.g., privacy concerns. This has motivated the graph-kernel based approaches in [186] and [189], to reconstruct the graph signal when only the network structure is available, and the kernel matrix is selected as a nonlinear function of the graph Laplacian matrix. Specifically, these works mostly consider undirected networks, $\mathbf{A} = \mathbf{A}^\top$.

Given the normalized Laplacian matrix $\mathbf{L} := \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$, with $\mathbf{D} := \mathrm{diag}(\mathbf{A}\mathbf{1})$, and letting $\mathbf{L} := \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^\top$, the family of graphical kernels is

$$\bar{\mathbf{K}} := r^\dagger(\mathbf{L}) := \mathbf{U} r^\dagger(\boldsymbol{\Lambda}) \mathbf{U}^\top \tag{6.7}$$

where $r(.)$ is a non-decreasing scalar function of the eigenvalues, and † denotes pseudo-inverse. By selecting $r(.)$, different graph properties can be accounted for, including smoothness, band-limitedness, the random walk [189], and diffusion [175].

Although graph-kernel based methods are effective in reconstructing signals over graphs, it can be observed from (6.7) that formulating $\bar{\mathbf{K}}$ generally requires an eigenvalue decomposition of $\mathbf{L}$, which incurs complexity $\mathcal{O}(N^3)$ that can be prohibitive for large-scale networks. Moreover, even though nodal feature vectors $\{\phi_n\}$ are not necessary to form $\bar{\mathbf{K}}$, the graph-kernel-based scheme requires knowledge of the topology, meaning $\mathbf{A}$, in order to estimate the nodal function of each node. However, in networks with strict privacy requirements, nodes may not be willing to share such information with others. In Facebook, for example, most people do not make their friend list public. In addition, solving (6.4) assumes that all sampled nodes are available in batch, which may not be true in scenarios where nodes are sampled in a sequential fashion.

In response to these challenges, an online scalable kernel-based method will be developed in the ensuing section to deal with sequentially obtained data samples, over generally dynamic networks. The resultant algorithm only requires encrypted versions of the nodal connectivity patterns of other nodes, and hence it offers privacy.

6.3 Online kernel-based learning over graphs

Instead of resorting to a graph kernel that requires an eigenvalue decomposition of $\mathbf{L}$ in (6.7), the present section advocates treating the *connectivity pattern of each node as its feature vector*, which can be the nth column $\mathbf{a}_n^{(c)}$ and possibly the nth row $(\mathbf{a}_n^{(r)})^\top$ of the adjacency (if $\mathbf{A}$ is nonsymmetric). We will henceforth term this the *connectivity pattern* of v_n, and denote it as $\mathbf{a}_n$, for brevity. Given $\mathbf{a}_n$, we will interpolate unavailable nodal function values $\hat{f}(v_n)$ using a nonparametric approach, that is different and scalable relative to [189] and [186]. The kernel matrix is now

$$[\bar{\mathbf{K}}]_{n,n'} = \kappa(v_n, v_{n'}) = \kappa(\mathbf{a}_n, \mathbf{a}_{n'}). \tag{6.8}$$

Again, with M nodes sampled, the representer theorem asserts that the sought function estimator has the form [70]

$$\hat{f}(v_n) = \hat{f}(\mathbf{a}_n) = \sum_{m=1}^{M} \alpha_m \kappa(\mathbf{a}_m, \mathbf{a}_n) := \boldsymbol{\alpha}^\top \mathbf{k}(\mathbf{a}_n) \tag{6.9}$$

where $\mathbf{k}(\mathbf{a}_n) := [\kappa(\mathbf{a}_n, \mathbf{a}_1) \ldots \kappa(\mathbf{a}_n, \mathbf{a}_M)]^\top$. It can be observed from (6.9) that $\hat{f}(v_n)$ involves the adjacency of the entire network, namely $\{\mathbf{a}_m\}_{m=1}^M$, which leads to potentially growing complexity $\mathcal{O}(M^3)$ as the number of sampled nodes increases [70].

6.3.1 Batch RF-based learning over graphs

To bypass this growing complexity, we will resort to the so-called random feature approximation [75] in order to reduce the original functional learning task in (6.4) to a problem with the number of unknown parameters not growing with M. We first approximate κ in (6.5) using random features (RFs) [13, 75] that are obtained from a shift-invariant kernel satisfying $\kappa(\mathbf{a}_n, \mathbf{a}_{n'}) = \kappa(\mathbf{a}_n - \mathbf{a}_{n'})$. For $\kappa(\mathbf{a}_n - \mathbf{a}_{n'})$ absolutely integrable, its Fourier transform $\pi_\kappa(\mathbf{v})$ exists and represents the power spectral density, which upon normalizing to ensure $\kappa(\mathbf{0}) = 1$, can also be viewed as a probability density function (pdf); hence,

$$\kappa(\mathbf{a}_n - \mathbf{a}_{n'}) = \int \pi_\kappa(\mathbf{v}) e^{j\mathbf{v}^\top(\mathbf{a}_n - \mathbf{a}_{n'})} d\mathbf{v} := \mathbb{E}_{\mathbf{v}}\left[e^{j\mathbf{v}^\top(\mathbf{a}_n - \mathbf{a}_{n'})}\right] \tag{6.10}$$

where the last equality is due to the definition of the expected value. Drawing a sufficient number of D independent and identically distributed samples $\{\mathbf{v}_i\}_{i=1}^D$ from $\pi_\kappa(\mathbf{v})$, the ensemble mean (6.10) can be approximated by the sample average

$$\hat{\kappa}(\mathbf{a}_n, \mathbf{a}_{n'}) = \mathbf{z}_{\mathbf{V}}^\top(\mathbf{a}_n)\mathbf{z}_{\mathbf{V}}(\mathbf{a}_{n'}) \tag{6.11}$$

where $\mathbf{V} := [\mathbf{v}_1, \ldots, \mathbf{v}_D]^\top \in \mathbb{R}^{D \times N}$, and $\mathbf{z}_{\mathbf{V}}$ denotes the $2D \times 1$ *real-valued* RF vector

$$\begin{aligned}\mathbf{z}_{\mathbf{V}}(\mathbf{a}) = {} & D^{-\frac{1}{2}} \\ & \times \left[\sin(\mathbf{v}_1^\top \mathbf{a}), \ldots, \sin(\mathbf{v}_D^\top \mathbf{a}), \cos(\mathbf{v}_1^\top \mathbf{a}), \ldots, \cos(\mathbf{v}_D^\top \mathbf{a})\right]^\top.\end{aligned} \tag{6.12}$$

Taking expectations in (6.11) and using (6.10), one can verify that $\mathbb{E}_{\mathbf{v}}[\hat{\kappa}(\mathbf{a}_n, \mathbf{a}_{n'})] = \kappa(\mathbf{a}_n, \mathbf{a}_{n'})$, which means $\hat{\kappa}$ is unbiased. Note that finding $\pi_\kappa(\mathbf{v})$ requires an N-dimensional Fourier transform of κ, which in general requires numerical integration. Nevertheless, it has been shown that for a number of popular kernels, $\pi_\kappa(\mathbf{v})$ is available in closed form [75]. Taking the Gaussian kernel as an example, where $\kappa(\mathbf{a}_n, \mathbf{a}_{n'}) = \exp\left(\|\mathbf{a}_n - \mathbf{a}_{n'}\|_2^2/(2\sigma^2)\right)$, it has a Fourier transform corresponding to the pdf $\mathcal{N}(0, \sigma^{-2}\mathbf{I})$.

Hence, the function that is optimal in the sense of (6.3) can be cast to a function approximant over the $2D$-dimensional RF space (cf. (6.9) and (6.11))

$$\hat{f}^{\mathrm{RF}}(\mathbf{a}) = \sum_{m=1}^{M} \alpha_m \mathbf{z}_{\mathbf{V}}^\top(\mathbf{a}_m)\mathbf{z}_{\mathbf{V}}(\mathbf{a}) := \boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}) \tag{6.13}$$

where $\boldsymbol{\theta}^\top := \sum_{m=1}^{M} \alpha_m \mathbf{z}_{\mathbf{V}}^\top(\mathbf{a}_m)$. While $\hat{f}$ in (6.5) is the superposition of nonlinear functions κ, its RF approximant $\hat{f}^{\mathrm{RF}}$ in (6.13) is a linear function of $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_i)$. As a result, (6.3) reduces to

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{2D}} \frac{1}{M}\sum_{m=1}^{M} \mathcal{C}(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_m), y_m) + \mu\Omega\left(\|\boldsymbol{\theta}\|^2\right) \tag{6.14}$$

where $\|\boldsymbol{\theta}\|^2 := \sum_t \sum_\tau \alpha_t \alpha_\tau \mathbf{z}_{\mathbf{V}}^\top(\mathbf{a}_t)\mathbf{z}_{\mathbf{V}}(\mathbf{a}_\tau) := \|f\|_{\mathcal{H}}^2$. A batch solver of (6.14) has complexity $\mathcal{O}(MD^3)$ that does not grow with N. This batch RF-based approach scales linearly with the number of measured nodes M, and the number of variables is $2D$, which does not depend on M. This allows us to pursue an online implementation as elaborated next.

6.3.2 Online RF-based learning over graphs

Here, we will further leverage RF-based learning over graphs to enable real-time learning and reconstruction of signals evolving over possibly dynamic networks. A scalable online algorithm will be introduced, which can adaptively handle sequentially sampled nodal features and update the sought function estimates.

Training sequentially. In the training phase, we are given a network of N nodes, and the nodal function is sampled in a sequential fashion. Letting v_t denote the node sampled at the tth time slot, and having available $\{\mathbf{a}_t, y_t\}$ at v_t, the online inference task can be written as [cf. (6.14)]

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{2D}} \sum_{\tau=1}^{t} \mathcal{L}\left(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_\tau), y_\tau\right) \tag{6.15}$$

$$\mathcal{L}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_t), y_t\big) := \mathcal{C}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_t), y_t\big) + \mu\Omega\big(\|\boldsymbol{\theta}\|^2\big).$$

We will solve (6.15) using online gradient descent [89]. Obtaining v_t per slot t, the RF of its connectivity pattern $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t)$ is formed as in (6.12), and $\boldsymbol{\theta}_{t+1}$ is updated 'on the fly,' as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla\mathcal{L}(\boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_t), y_t) \tag{6.16}$$

where $\{\eta_t\}$ is the sequence of stepsizes that can tune learning rates. In this chapter, we will adopt $\eta_t = \eta$ for simplicity. Iteration (6.16) provides *a functional update* since $\hat{f}_t^{\mathrm{RF}}(\mathbf{a}) = \boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a})$. The per-iteration complexity of (6.16) is $\mathcal{O}(D)$, and $\mathcal{O}(MD)$ for the entire training process, which scales better than $\mathcal{O}(MD^3)$ that is required for a batch solver of (6.14).

Inferring unavailable nodal values. After the training phase, the nodal function value over the un-sampled nodes can be readily estimated by [cf. (6.13)]

$$\hat{f}(v_i) = \hat{\boldsymbol{\theta}}^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_i), \;\; \forall i \in \mathcal{S}^c \tag{6.17}$$

where $\hat{\boldsymbol{\theta}}$ is the final estimate after the training phase, i.e., $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{M+1}$, and $\mathcal{S}^c$ denotes the index set of the nodes whose signal values have not been sampled in the training phase.

Newly-joining nodes. When new nodes join the network, batch graph-kernel based approaches must expand $\bar{\mathbf{K}}$ in (6.7) by one row and one column, and re-solve (6.6) in order to form signal estimates for the newly-joining nodes. Hence, each newly joining node will incur complexity $\mathcal{O}(N^3)$. The novel online RF method on the other hand, can simply estimate the signal on the newly coming node via $\hat{f}(v_{\text{new}}) = \hat{\boldsymbol{\theta}}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_{\text{new}})$, where $\mathbf{a}_{\text{new}} \in \mathbb{R}^N$ denotes the connectivity pattern of the new node with the *existing* nodes in the network. This leads to a complexity of $\mathcal{O}(ND)$ per new node. If in addition, y_{new} is available, then the function estimate can also be efficiently updated via (6.16) and (6.13) using $\mathbf{a}_{\text{new}}$ and y_{new}. The steps of our online RF-based method are summarized in Algorithm 7b.

Remark 1 (Privacy). Note that the update in (6.16) does not require access to $\mathbf{a}_t$ directly. Instead, the only information each node needs to reveal is $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t)$ for each $\mathbf{a}_t$, which involves $\{\sin(\mathbf{a}_t^\top \mathbf{v}_j), \;\; \cos(\mathbf{a}_t^\top \mathbf{v}_j)\}_{j=1}^D$. Being noninvertible, these co-sinusoids functions involved in generating the $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t)$ can be viewed as an encryption of the nodal connectivity pattern, which means that given $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t)$, vector $\mathbf{a}_t$ cannot be uniquely deciphered. Hence, Algorithm 7b preserves privacy.

Remark 2 (Directed graphs). It can be observed from (6.7) that for $\bar{\mathbf{K}}$ to be a valid kernel, graph-kernel based methods require $\mathbf{A}$, and henceforth $\mathbf{L}$ to be symmetric, which implies they can only directly deal with symmetric/undirected graphs. Such a requirement is not necessary for our RF-based method.

Remark 3 (Dynamic graphs). Real-world networks may vary over time, as edges may disappear or appear. To cope with such changing topologies, the original graph-kernel method needs to recalculate the kernel matrix, and resolve the batch problem whenever one edge changes. In contrast, our online RF-based method can simply re-estimate the nodal values on the two ends of the (dis)appeared edge using (6.13) with their current $\{\mathbf{a}_n\}$.

Evidently, the performance of Algorithm 7b depends on κ that is so far considered known.

Algorithm 7b Online kernel based learning over graphs

```
1: **Input:** step size $\eta > 0$, and number of RFs $D$.
2: **Initialization:** $\boldsymbol{\theta}_1 = \mathbf{0}$.
3: **Training:**
4: **for** $t = 1, 2, \ldots, M$ **do**
5:     Obtain the adjacency vector $\mathbf{a}_t$ of sampled node $v_t$ .
6:     Construct $\mathbf{z}_p(\mathbf{a}_t)$ via (6.12) using $\kappa$.
7:     Update $\boldsymbol{\theta}_{t+1}$ via (6.16).
8: **end for**
9: **Inference:**
10:    Construct random feature vector $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_j)$ via (6.12)
11:    Infer $\hat{f}(v_j) = \boldsymbol{\theta}_{M+1}^{\top}\mathbf{z}_{\mathbf{V}}(v_j), \ \ j \in \Omega$.
12: **Accounting for newly-coming node**
13:    Construct random feature vector $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_{\text{new}})$ via (6.12)
14:    Estimate $\hat{f}(v_{\text{new}}) = \boldsymbol{\theta}_{M+1}^{\top}\mathbf{z}_{\mathbf{V}}(v_{\text{new}})$.
15:    If $y_{\text{new}}$ available, Update $\boldsymbol{\theta}$ via (6.16).
```

As the "best" performing κ is generally unknown and application dependent, it is prudent to adaptively select kernels by superimposing multiple kernels from a prescribed dictionary, as we elaborate next.

6.4 Online Graph-adaptive MKL

In the present section, we develop an online **gr**aph-**ad**aptive learning approach that relies on **ra**ndom features, and leverages multi-**ker**nel approximation to estimate the desired f based on sequentially obtained nodal samples over the graph. The proposed method is henceforth abbreviated as **Gradraker**.

The choice of κ is critical for the performance of single kernel based learning over graphs, since different kernels capture different properties of the graph, and thus lead to function estimates of variable accuracy [186]. To deal with this, combinations of kernels from a preselected dictionary $\{\kappa_p\}_{p=1}^{P}$ can be employed in (6.3); see also [13, 186]. Each combination belongs to the convex hull $\bar{\mathcal{K}} := \{\bar{\kappa} = \sum_{p=1}^{P} \bar{\alpha}_p \kappa_p, \ \bar{\alpha}_p \geq 0, \ \sum_{p=1}^{P} \bar{\alpha}_p = 1\}$. With $\bar{\mathcal{H}}$ denoting the RKHS induced by $\bar{\kappa} \in \bar{\mathcal{K}}$, one then solves (6.3) with $\mathcal{H}$ replaced by $\bar{\mathcal{H}} := \mathcal{H}_1 \bigoplus \cdots \bigoplus \mathcal{H}_P$, where $\{\mathcal{H}_p\}_{p=1}^{P}$ represent the RKHSs corresponding to $\{\kappa_p\}_{p=1}^{P}$ [63].

The candidate function $\bar{f} \in \bar{\mathcal{H}}$ is expressible in a separable form as $\bar{f}(\mathbf{a}) := \sum_{p=1}^{P} \bar{f}_p(\mathbf{a})$, where $\bar{f}_p(\mathbf{a})$ belongs to $\mathcal{H}_p$, for $p \in \mathcal{P} := \{1, \ldots, P\}$. To add flexibility per kernel in our ensuing online MKL scheme, we let wlog $\{\bar{f}_p = w_p f_p\}_{p=1}^P$, and seek functions of the form

$$f(v) = f(\mathbf{a}) := \sum_{p=1}^{P} \bar{w}_p f_p(\mathbf{a}) \in \bar{\mathcal{H}} \tag{6.18}$$

where $f := \bar{f} / \sum_{p=1}^P w_p$, and the normalized weights $\{\bar{w}_p := w_p / \sum_{p=1}^P w_p\}_{p=1}^P$ satisfy $\bar{w}_p \geq 0$, and $\sum_{p=1}^P \bar{w}_p = 1$. Given the connectivity pattern $\mathbf{a}_t$ of the tth sampled node v_t, an RF vector $\mathbf{z}_p(\mathbf{a}_t)$ is generated per p from the pdf $\pi_{\kappa_p}(\mathbf{v})$ via (6.12), where $\mathbf{z}_p(\mathbf{a}_t) := \mathbf{z}_{\mathbf{V}_p}(\mathbf{a}_t)$ for notational brevity. Hence, per kernel κ_p and node sample t, we have [cf. (6.13)]

$$\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{a}_t) = \boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{a}_t). \tag{6.19}$$

Letting $\mathcal{L}_t(\hat{f}_p^{\mathrm{RF}}(\mathbf{a}_t)) := \mathcal{L}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}_p}(\mathbf{a}_t), y_t\big)$ in (6.15), and as in (6.16), $\boldsymbol{\theta}_{p,t}$ is updated via

$$\boldsymbol{\theta}_{p,t+1} = \boldsymbol{\theta}_{p,t} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{a}_t), y_t) \tag{6.20}$$

with $\eta \in (0, 1)$ chosen constant to effect the adaptation. As far as $\bar{w}_{p,t}$ is concerned, since it resides on the probability simplex, a multiplicative update is well motivated as discussed also in, e.g., [13, 89]. For the un-normalized weights, this update is available in closed form as [13]

$$w_{p,t+1} = w_{p,t} \exp\left(-\eta \mathcal{L}_t\left(\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{a}_t)\right)\right). \tag{6.21}$$

Having found $\{w_{p,t}\}$ as in (6.21), the normalized weights in (6.18) are obtained as $\bar{w}_{p,t} := w_{p,t} / \sum_{p=1}^P w_{p,t}$. Note from (6.21) that when $\hat{f}_{p,t}^{\mathrm{RF}}$ has a larger loss relative to other $\hat{f}_{p',t}^{\mathrm{RF}}$ with $p' \neq p$ for the tth sampled node, the corresponding $w_{p,t+1}$ decreases more than the other weights. In other words, a more accurate approximant tends to play a more important role in predicting the ensuing sampled node. In summary, our Gradraker for online graph MKL is listed as Algorithm 8b.

Remark 4 (Comparison with batch MKL). A batch MKL based approach for signal reconstruction over graphs was developed in [186]. It entails an iterative algorithm whose complexity grows with N in order to jointly estimate the nodal function, and to adaptively select the kernel function. When new nodes join the network, [186] re-calculates the graphical kernels and re-solves the overall batch problem, which does not scale with the network size. In addition, [186]

Algorithm 8b Gradraker algorithm

```
1: Input: Kernels $\kappa_p,\ p = 1, \ldots, P$, step size $\eta > 0$, and number of RFs $D$.
2: Initialization: $\boldsymbol{\theta}_{p,1} = \mathbf{0}$.
3: Training:
4: for $t = 1, 2, \ldots, T$ do
5:     Obtain the adjacency vector $\mathbf{a}_t$ of node $v_t$ .
6:     Construct $\mathbf{z}_p(\mathbf{a}_t)$ via (6.12) using $\kappa_p$ for $p = 1, \ldots, P$.
7:     Predict $\hat{f}^{\mathrm{RF}}_t(\mathbf{a}_t) = \sum_{p=1}^{P} \bar{w}_{p,t} \hat{f}^{\mathrm{RF}}_{p,t}(\mathbf{a}_t)$
8:     Observe loss function $\mathcal{L}_t$, incur $\mathcal{L}_t(\hat{f}^{\mathrm{RF}}_t(\mathbf{a}_t))$.
9:     for $p = 1, \ldots, P$ do
10:        Obtain loss $\mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{a}_t), y_t)$ or $\mathcal{L}_t(\hat{f}^{\mathrm{RF}}_{p,t}(\mathbf{a}_t))$.
11:        Update $\boldsymbol{\theta}_{p,t+1}$ and $w_{p,t+1}$ via (6.20) and (6.21).
12:    end for
13: end for
14: Inference:
15:    Construct RF vector $\{\mathbf{z}_p(\mathbf{a}_j)\}$ using $\{\kappa_p\}$.
16:    Infer $\hat{f}(v_j) = \sum_{p=1}^{P} \bar{w}_{p,T+1} \boldsymbol{\theta}_{p,T+1}^\top \mathbf{z}_p(v_j)$.
17: Accounting for newly-coming node
18:    Construct RF vector $\{\mathbf{z}_p(\mathbf{a}_{\mathrm{new}})\}$ using $\{\kappa_p\}$.
19:    Estimate $\hat{f}(v_{\mathrm{new}}) = \sum_{p=1}^{P} \bar{w}_{p,T+1} \boldsymbol{\theta}_{p,T+1}^\top \mathbf{z}_p(v_{\mathrm{new}})$.
20:    If $y_{\mathrm{new}}$ available update $\{\boldsymbol{\theta}_p, w_p\}$ via (6.20) and (6.21).
```

is not privacy preserving in the sense that in order to estimate the function at any node, one needs to have access to the connectivity pattern of the entire network.

Remark 5 (Comparison with k-NN). An intuitive yet efficient way to predict function values of a newly joining node is to simply combine the values of its k nearest neighbors (k-NN) [192, 193]. Efficient as it is, k-NN faces several challenges: a) At least one of the neighbors must be labeled, which does not always hold in practice, and is not required by the Gradraker; and b) k-NN can only account for local information, while the Gradraker takes also into account the global information of the graph.

6.4.1 Generalizations

So far, it is assumed that each node n only has available its own connectivity feature vector $\mathbf{a}_n$. This allows Gradraker to be applied even when limited information is available about the nodes, which many existing algorithms that rely on nodal features cannot directly cope with.

If additional feature vectors $\{\boldsymbol{\phi}_{i,n}\}_{i=1}^{I}$ are actually available per node n other than its own

$\mathbf{a}_n$, it is often not known a priori which set of features is the most informative for estimating the signal of interest on the graph. To this end, the novel Gradraker can be adapted by treating the functions learned from different sets of features as *an ensemble of learners*, and combine them in a similar fashion as in (6.18), that is,

$$f(v_n) = \sum_{i=1}^{I} \beta_i f_i(\boldsymbol{\phi}_{i,n}) \tag{6.22}$$

Applications to several practical scenarios are discussed in the following.

Semi-private networks. In practice, a node may tolerate sharing its links to its neighbors, e.g., users of Facebook may share their friends-list with friends. In this scenario, each node not only knows its own neighbors, but also has access to who are its neighbors' neighbors, i.e., two-hop neighbors. Specifically, node n has access to $\mathbf{a}_n$, as well as to the nth column of $\mathbf{A}^{(2)} := \mathbf{A}\mathbf{A}$ [9], and a learner $f_2(\boldsymbol{\phi}_{2,n})$ can henceforth be introduced and combined in (6.22). Moreover, when nodes are less strict about privacy, e.g., when a node is willing to share its multi-hop neighbors, more learners can be introduced and combined 'on the fly' by selecting $\boldsymbol{\phi}_{i,n}$ as the nth column of $\mathbf{A}^{(i)}$ in (6.22).

Multilayer networks. Despite their popularity, ordinary networks are often inadequate to describe increasingly complex systems. For instance, modeling interactions between two individuals using a single edge can be a gross simplification of reality. Generalizing their *single-layer* counterparts, *multilayer networks* allow nodes to belong to N_g groups, called layers [194, 195]. These layers could represent different attributes or characteristics of a complex system, such as temporal snapshots of the same network, or different types of groups in social networks (family, soccer club, or work related). Furthermore, multilayer networks are able to model systems that typically cannot be represented by traditional graphs, such as heterogeneous information networks [196, 197]. To this end, Gradraker can readily incorporate the information collected from heterogenous sources, e.g., connectivity patterns $\{\mathbf{A}_i\}_{i=1}^{N_g}$ from different layers, by adopting a kernel based learner $f_i(\mathbf{a}_{i,n})$ on the ith layer and combining them as in (6.22).

Nodal features available. In certain cases, nodes may have nodal features [9] in addition to their $\{\mathbf{a}_n\}$. For example, in social networks, other than the users' connectivity patterns, we may also have access to their shopping history on Amazon. In financial networks, in addition to the knowledge of trade relationships with other companies, there may be additional information available per company, e.g., the number of employees, category of products the company sales, or the annual profit. Gradraker can also incorporate this information by introducing additional learners based on the nodal feature vectors, and combine them as in (6.22).

6.5 Performance analysis

To analyze the performance of the novel Gradraker algorithm, we assume that the following are satisfied.

(as1) *For all sampled nodes $\{v_t\}_{t=1}^T$, the loss function $\mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_\mathbf{V}(\mathbf{a}_t), y_t)$ in* (6.15) *is convex w.r.t. $\boldsymbol{\theta}$.*

(as2) *For $\boldsymbol{\theta}$ belonging to a bounded set $\boldsymbol{\Theta}$ with $\|\boldsymbol{\theta}\| \leq C_\theta$, the loss is bounded; that is, $\mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_\mathbf{V}(\mathbf{a}_t), y_t) \in [-1, 1]$, and has bounded gradient, meaning, $\|\nabla \mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_\mathbf{V}(\mathbf{a}_t), y_t)\| \leq L$.*

(as3) *The kernels $\{\kappa_p\}_{p=1}^P$ are shift-invariant, standardized, and bounded, that is, $\kappa_p(\mathbf{a}_n, \mathbf{a}_{n'}) \leq 1$, $\forall \mathbf{a}_n, \mathbf{a}_{n'}$; and w.l.o.g. they also have bounded entries, meaning $\|\mathbf{a}_n\| \leq 1, \forall n$.*

Convexity of the loss under (as1) is satisfied by the popular loss functions including the square loss and the logistic loss. As far as (as2), it ensures that the losses, and their gradients are bounded, meaning they are L-Lipschitz continuous. While boundedness of the losses commonly holds since $\|\boldsymbol{\theta}\|$ is bounded, Lipschitz continuity is also not restrictive. Considering kernel-based regression as an example, the gradient is $(\boldsymbol{\theta}^\top \mathbf{z}_\mathbf{V}(\mathbf{x}_t) - y_t)\mathbf{z}_\mathbf{V}(\mathbf{x}_t) + \lambda\boldsymbol{\theta}$. Since the loss is bounded, e.g., $\|\boldsymbol{\theta}^\top \mathbf{z}_\mathbf{V}(\mathbf{x}_t) - y_t\| \leq 1$, and the RF vector in (6.12) can be bounded as $\|\mathbf{z}_\mathbf{V}(\mathbf{x}_t)\| \leq 1$, the constant is $L := 1 + \lambda C_\theta$ using the Cauchy-Schwartz inequality. Kernels satisfying the conditions in (as3) include Gaussian, Laplacian, and Cauchy [75]. In general, (as1)-(as3) are standard in online convex optimization (OCO) [53, 89], and in kernel-based learning [75, 86, 90].

In order to quantify the performance of Gradraker, we resort to the static regret metric, which quantifies the difference between the aggregate loss of an OCO algorithm, and that of the best fixed function approximant in hindsight, see also e.g., [53, 89]. Specifically, for a sequence $\{\hat{f}_t\}$ obtained by an online algorithm $\mathcal{A}$, its static regret is

$$\mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}}(T) := \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_t(\mathbf{a}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(f^*(\mathbf{a}_t)) \tag{6.23}$$

where $\hat{f}_t^{\mathrm{RF}}$ will henceforth be replaced by $\hat{f}_t$ for notational brevity; and, $f^*(\cdot)$ is defined as the batch solution

$$f^*(\cdot) \in \arg \min_{\{f_p^*,\, p\in\mathcal{P}\}} \sum_{t=1}^{T} \mathcal{L}_t(f_p^*(\mathbf{a}_t))$$
$$\text{with} \quad f_p^*(\cdot) \in \arg \min_{f\in\mathcal{F}_p} \sum_{t=1}^{T} \mathcal{L}_t(f(\mathbf{a}_t)) \tag{6.24}$$

where $\mathcal{F}_p := \mathcal{H}_p$, with $\mathcal{H}_p$ representing the RKHS induced by κ_p. We establish the regret of our Gradraker approach in the following lemma.

Lemma 11: *Under (as1), (as2), and with $\hat{f}_p^*$ defined as $\hat{f}_p^*(\cdot) \in \arg\min_{f\in\hat{\mathcal{F}}_p} \sum_{t=1}^{T} \mathcal{L}_t(f(\mathbf{a}_t))$, with $\hat{\mathcal{F}}_p := \{\hat{f}_p | \hat{f}_p(\mathbf{a}) = \boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{a}),\, \forall \boldsymbol{\theta} \in \mathbb{R}^{2D}\}$, for any p, the sequences $\{\hat{f}_{p,t}\}$ and $\{\bar{w}_{p,t}\}$ generated by Gradraker satisfy the following bound*

$$\sum_{t=1}^{T} \mathcal{L}_t\bigg(\sum_{p=1}^{P} \bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{a}_t)\bigg) - \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_p^*(\mathbf{a}_t))$$
$$\leq \frac{\ln P}{\eta} + \frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta} + \frac{\eta L^2 T}{2} + \eta T \tag{6.25}$$

where $\boldsymbol{\theta}_p^*$ is associated with the best RF function approximant $\hat{f}_p^*(\mathbf{a}) = \left(\boldsymbol{\theta}_p^*\right)^\top \mathbf{z}_p(\mathbf{a})$.

Proof: See Appendix 3.7.1

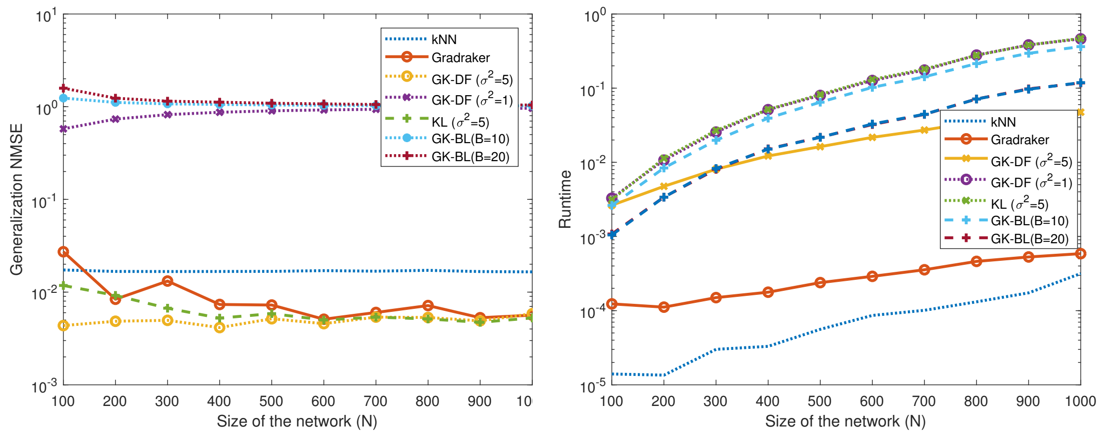
In addition to bounding the regret in the RF space, the next theorem compares the Gradraker loss relative to that of the best functional estimator in the original RKHS.

Theorem 7 *Under (as1)-(as3), and with f^* defined as in (6.24), for a fixed $\epsilon > 0$, the following bound holds with probability at least $1 - 2^8\left(\frac{\sigma_p}{\epsilon}\right)^2 \exp\left(\frac{-D\epsilon^2}{4N+8}\right)$*

$$\sum_{t=1}^{T} \mathcal{L}_t\left(\sum_{p=1}^{P} \bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{a}_t)\right) - \sum_{t=1}^{T} \mathcal{L}_t\left(f^*(\mathbf{a}_t)\right)$$
$$\leq \frac{\ln P}{\eta} + \frac{(1+\epsilon)C^2}{2\eta} + \frac{\eta L^2 T}{2} + \eta T + \epsilon L T C \tag{6.26}$$

where C is a constant, while $\sigma_p^2 := \mathbb{E}_{\pi_{\kappa_p}}[\|\mathbf{v}\|^2]$ is the second-order moment of the RF vector norm. Setting $\eta = \epsilon = \mathcal{O}(1/\sqrt{T})$ in (6.26), the static regret in (6.23) leads to

$$\text{Reg}^{\text{s}}_{\text{Gradraker}}(T) = \mathcal{O}(\sqrt{T}). \tag{6.27}$$



(a) Generalization NMSE (b) Testing runtime

Figure 6.1: Inference performance versus number of nodes for synthetic dataset generated from graph diffusion kernel

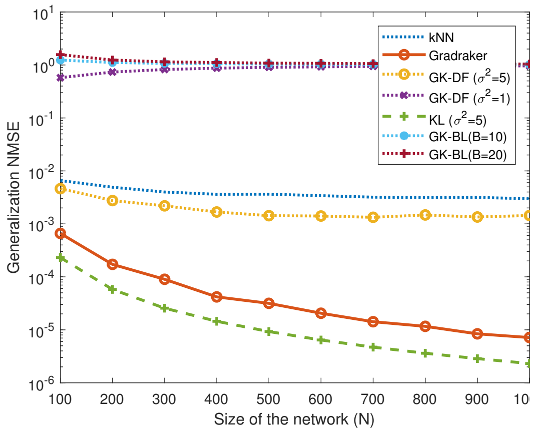
Proof: See Appendix 3.7.2

Observe that the probability of (6.26) to hold grows as D increases, and one can always find a D to ensure a positive probability for a given ϵ. Theorem 7 establishes that with a proper choice of parameters, the Gradraker achieves sub-linear regret relative to the best static function approximant in (6.24), which means the novel Gradraker algorithm is capable of capturing the nonlinear relationship among nodal functions accurately, as long as enough nodes are sampled sequentially.

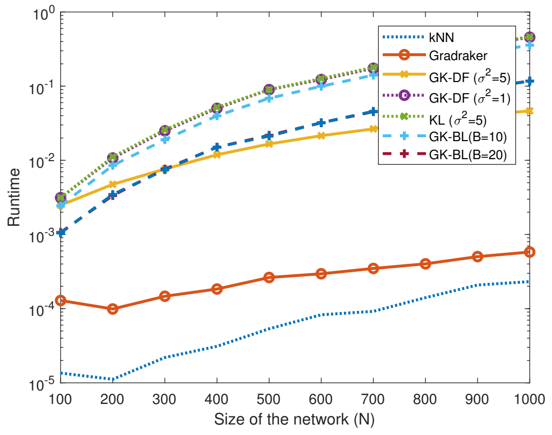
In addition, it is worth noting that Theorem 7 holds true regardless of the sampling order of the nodes $\{v_1, \ldots, v_T\}$. However, optimizing over the sampling pattern is possible, and constitutes one of our future research directions.

6.6 Numerical tests

In this section, Gradraker is tested on both synthetic and real datasets to corroborate its effectiveness. The tests will mainly focus on regression tasks for a fair comparison with existing alternatives.



(a) Generalization NMSE

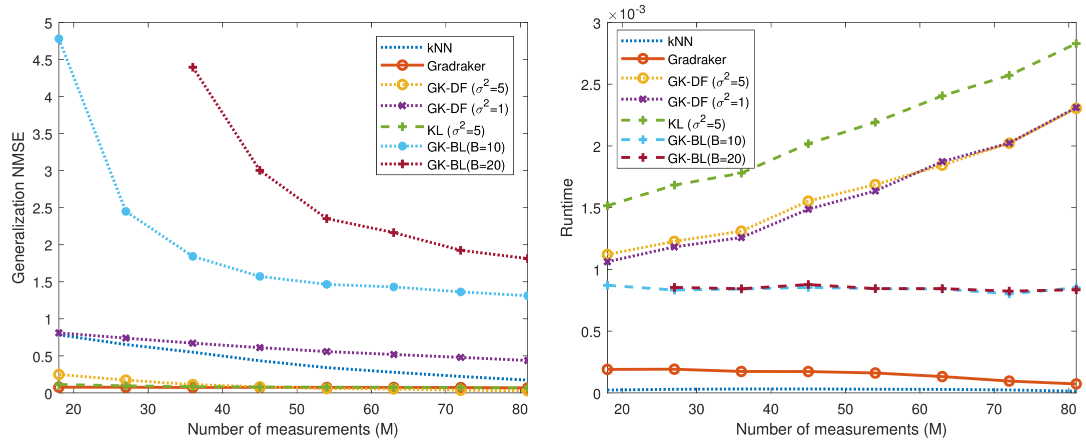


(b) Runtime

Figure 6.2: Inference performance versus number of nodes for synthetic dataset generated from Gaussian kernel

6.6.1 Synthetic data test

Data generation. An Erdös-Rényi graph [198] with binary adjacency matrix $\mathbf{A}_0 \in \mathbf{R}^{N\times N}$ was generated with probability of edge presence $\pi = 0.2$, and its adjacency was symmetrized as $\mathbf{A} = \mathbf{A}_0 + \mathbf{A}_0^\top$. This symmetrization is not required by Gradraker, but it is necessary for alternative graph kernel based methods. A function over this graph was then generated with each entry of the coefficient vector $\boldsymbol{\alpha} \in \mathbb{R}^N$ drawn uniformly from $[0.5, 1]$, and each entry of the noise $\mathbf{e}$ drawn from $\mathcal{N}(0, 0.01\mathbf{I})$. In each experiment, the sampling matrix $\mathbf{\Psi}$ is randomly generated so that $M = 0.05N$ of the nodes are randomly sampled, and the remaining $N - M$ nodes are treated as newly-joining nodes, whose function values and connectivity patterns are both unknown at the training phase, and whose nodal function values are estimated based on their connectivity with existing nodes in the network during the testing phase. All algorithms are carried out on the training set of M nodes, and the obtained model is used to estimate the function value on the newly arriving nodes. The runtime for estimating the function value on the newly-joining nodes, as well as the generalization NMSE $:= \frac{1}{|\mathcal{S}^c|}\|\hat{\mathbf{x}}_{\mathcal{S}^c} - \mathbf{x}_{\mathcal{S}^c}\|_2^2/\|\mathbf{x}_{\mathcal{S}^c}\|_2^2$ performance is evaluated, with $\mathcal{S}^c$ denoting the index set of new nodes. The Gradraker adopts a dictionary consisting of 2 Gaussian kernels with parameters $\sigma^2 = 1, 5$, using $D = 10$ random features, and it is compared with: a) the kNN algorithm, with k selected as the maximum number of neighbors a node has in a specific network, and with the combining weights set to $1/k$



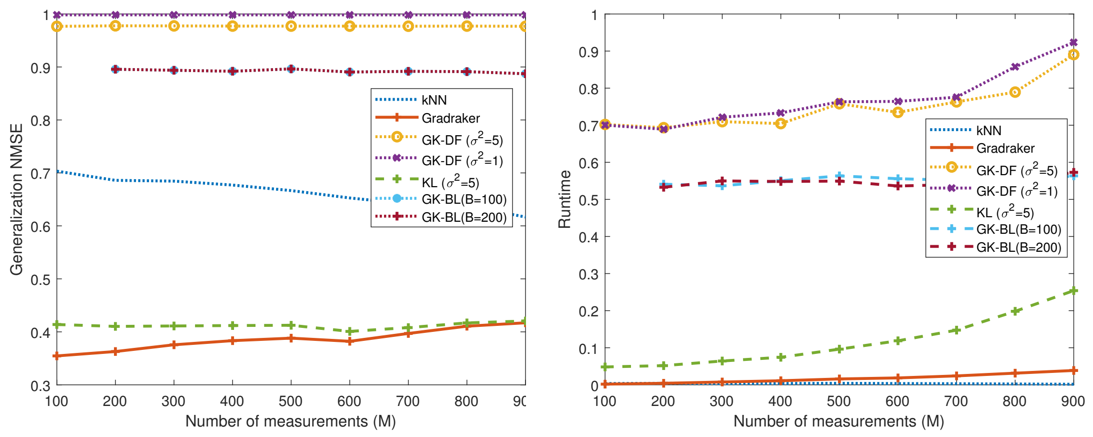
(a) Generalization NMSE (b) Runtime

Figure 6.3: Inference performance versus number of sampled nodes in temperature dataset

in unweighted graphs, and $a_{il}/\sum_{j\in\mathcal{N}_i} a_{ij}$ for the lth neighbor in weighted graphs; b) the graph kernel (GK) based method using diffusion kernels with different bandwidths (named as GK-DF), or band-limited kernels with different bandwidths (GK-BL); and c) kernel based learning without RF approximation (KL) with a Gaussian kernel of $\sigma^2 = 5$. Results are averaged over 100 independent runs. The regularization parameter for all algorithms is selected from the set $\mu = \{10^{-7}, 10^{-6}, \ldots, 10^{0}\}$ via cross validation.

Testing results. Figure 6.1 illustrates the performance in terms of the average runtime and NMSE versus the number of nodes (size) of the network. In this experiment, $\bar{\mathbf{K}}$ in (6.7) is generated from the normalized graph Laplacian $\mathbf{L}$, using the diffusion kernel $r(\lambda) = \exp(\sigma^2\lambda/2)$. A bandwidth of $\sigma^2 = 5$ was used to generate the data. It is observed that GK attains the best generalization accuracy when the ground-truth model is known, but its computational complexity grows rapidly with the network size. However, GK does not perform as well when a mismatched kernel is applied. The Gradraker method on the other hand, is very efficient, while at the same time it can provide reasonable estimates of the signal on the newly arriving nodes, even without knowledge about the kernels. The k-NN method is very efficient, but does not provide as reliable performance as the Gradraker.

Figure 6.2 depicts the performance of competitive algorithms. Matrix $\bar{\mathbf{K}}$ for data generation is formed based on (6.8) using the Gaussian kernel $\kappa(\mathbf{a}_i - \mathbf{a}_j) = \exp(\|\mathbf{a}_i - \mathbf{a}_j\|^2/\sigma^2)$, with $\sigma^2 = 5$. In this case, KL exactly matches the true model, and hence it achieves the best



(a) Generalization NMSE (b) Runtime

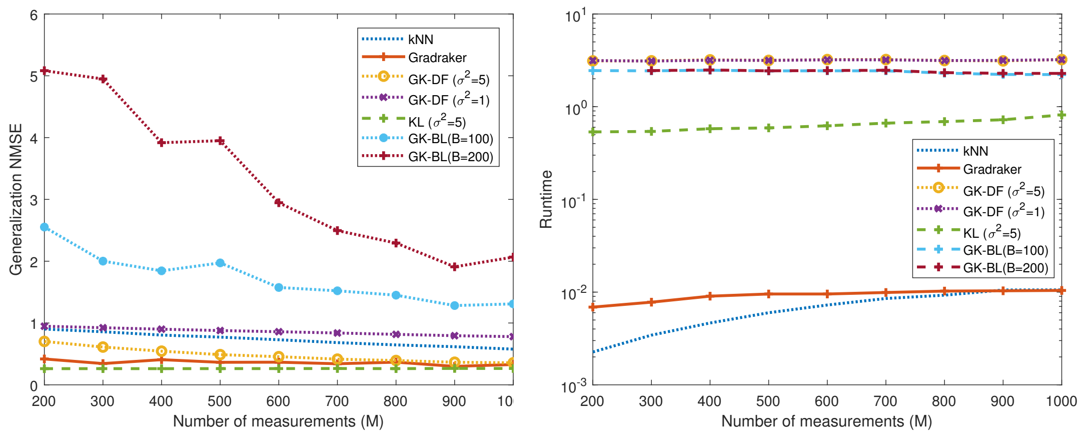
Figure 6.4: Inference performance versus number of sampled nodes in email dataset

performance. However, it is the most complex in terms of runtime. Meanwhile, GK-based methods suffer from model mismatch, and are also relatively more complex than Graderaker. The novel Gradraker is capable of estimating the nodal function on the newly joining nodes with high accuracy at very low computational complexity. Note that in real-world scenarios, accurate prior information about the underlying model is often unavailable, in which case Gradraker can be a more reliable and efficient choice.

6.6.2 Real data tests

Reconstruction of the temperature data. This subsection tests the performance of Gradraker on a real temperature dataset. The dataset comprises 24 signals corresponding to the average temperature per month in the intervals $1961 - 1980$ and $1991 - 2010$ measured by 89 stations in Switzerland [199]. The training set contains the first 12 signals, corresponding to the interval $1961 - 1980$, while the test set contains the remaining 12. Each station is represented by a node, and the graph was constructed using the algorithm in [200] based on the training signals. Given the test signal on a randomly chosen subset of M vertices, the values at the remaining $N - M$ vertices are estimated as newly-coming nodes. The generalization NMSE over the $N - M$ nodes is averaged across the test signals.

Fig. 6.3 compares the performance of Gradraker with those of competing alternatives. Gradraker adopts a dictionary consisting of 3 Gaussian kernels with parameters $\sigma^2 = 1, 5, 10$,



(a) Generalization NMSE (b) Runtime

Figure 6.5: Inference performance versus number of sampled nodes in Cora dataset

using $D = 100$ random features. It is clear from Fig. 6.3 that Gradraker outperforms GK in both generalization NMSE and runtime. On the other hand, even though KL achieves lower generalization NMSE, it incurs a much higher complexity.

Reconstruction of the Email-Eu-core data. The Eu-core network was generated using email data from a large European research institution [201], where each node represents a person, and an edge (i, j) is present if person i sent person j at least one email. The e-mails only represent communication between institution members (the core), and the dataset does not contain incoming messages from or outgoing messages to the rest of the world. The dataset also contains "ground-truth" community memberships of the nodes. Each individual belongs to one of 42 departments at the research institute. During the experiment, the department labels are considered to be y_n that are to be sampled and estimated. The graph consists of $N = 1,005$ nodes, and $25,571$ edges. Gradraker adopts a dictionary consisting of 2 Gaussian kernels with parameters $\sigma^2 = 1, 10$, from which $D = 10$ random features are generated. The test results were averaged over 100 independent runs with randomly sampled nodes.

Fig. 6.4 compares the performance of Gradraker with those of alternative algorithms when different numbers of nodal labels are observed. It is clear that the RF-based approach outperforms the GK-based method in both reconstruction accuracy and runtime. While the batch KL method without RF approximation outperforms the RF method, it incurs considerably higher computational complexity.

Reconstruction of the Cora data. This subsection tests the Gradraker algorithm on the Cora citation dataset [178]. Gradraker adopts a dictionary consisting of 2 Gaussian kernels with parameters $\sigma^2 = 1, 10$, using $D = 20$ random features. The results were averaged over 100 independent runs. The Cora dataset consists of $2, 708$ scientific publications classified into one of seven classes. The citation network consists of $5, 429$ links. The network is constructed so that a link connects node i to node j if paper i cites paper j, and the category id the paper belongs to is to be reconstructed. It can be observed again from Figure 6.5, that the Gradraker markedly outperforms the GK algorithms in terms of generalization NMSE, and is much more computationally efficient than all other algorithms except the kNN method, which however does not perform as well.

It can be readily observed from our numerical results over synthetic and real datasets, that the Gradraker provides reliable performance in terms of NMSE in all tests, while at the same time, it scales much better than all kernel based alternatives. This is because the alternative kernel-based algorithms require re-computing the kernel matrix whenever a new node joins the network. It is worth noting that all kernel-based alternatives require exact knowledge of the entire network topology, which is not necessary for GradRaker that only requires $\{\mathbf{z}_{\mathbf{V}}(\mathbf{a}_n)\}$. These tests corroborate the potential of GradRaker for application settings, where the graphs grow and nodes have privacy constraints.

6.7 Summary

The present chapter deals with the problem of reconstructing signals over graphs, from samples over a subset of nodes. An online MKL based algorithm is developed, which is capable of estimating and updating the nodal functions even when samples are collected sequentially. The novel online scheme is highly scalable and can estimate the unknown signals on newly joining nodes. Unlike many existing approaches, it only relies on encrypted nodal connectivity information, which is appealing for networks where nodes have strict privacy constraints.

This work opens up a number of interesting directions for future research, including: a) exploring distributed implementations that are well motivated in large-scale networks; b) graph-adaptive learning when multiple sets of features are available; and c) developing adaptive sampling strategies for Gradraker.

References

[1] I. T. Jolliffe, *Principal Component Analysis*. Springer, 2002.

[2] M. E. Tipping, “Probabilistic visualisation of high-dimensional binary data,” in *Proc. of Neural Info. Proc. Systems*, Denver, CO, USA, Dec. 1998, pp. 592–598.

[3] A. I. Schein, L. K. Saul, and L. H. Ungar, “A generalized linear model for principal component analysis of binary data,” in *Proc. Intl. Workshop on Artif. Intell. and Stat.*, vol. 38, Key West, FL, USA, Jan. 2003.

[4] S. Lee, J. Z. Huang, and J. Hu, “Sparse logistic principal components analysis for binary data,” *Ann. of Appl. Stat.*, vol. 4, no. 3, pp. 1579–1601, Oct. 2010.

[5] D. L. Jan, “Principal component analysis of binary data by iterated singular value decomposition,” *Comput. Stat. Data Anal.*, vol. 50, no. 1, pp. 21–39, 2006.

[6] L. Kozma, A. Ilin, and T. Raiko, “Binary principal component analysis in the netflix collaborative filtering task,” in *Proc. Intl. Workshop on Machine Learning for Signal Processing*, Grenoble, Sep. 2009, pp. 1–6.

[7] J. Mažgut, M. Paulinyová, and P. Tiňo, “Using dimensionality reduction method for binary data to questionnaire analysis,” in *Workshop on Math. and Eng. Methods in Comput. Sci.*, Lednice, Czech Republic, Oct. 2012, pp. 146–154.

[8] Z. Kang and C. J. Spanos, “Sequential logistic principal component analysis (SLPCA): Dimensional reduction in streaming multivariate binary-state system,” in *Intl. Conf. Mach. Learn. and App.*, Detroit, MI, USA, Dec. 2014, pp. 171–177.

[9] E. D. Kolaczyk, *Statistical Analysis of Network Data: Methods and Models*. Springer, 2009.

[10] Y. Shen, M. Mardani, and G. B. Giannakis, “Online sketching of big categorical data with absent features,” in *Proc. of Conf. on Info. Sciences and Systems*, Baltimore, MD, Mar. 2015.

[11] Y. Shen and G. B. Giannakis, “Online dictionary learning for large-scale binary data,” in *Proc. of Euro. Sig. Proc. Conf.*, Budapest, Hungary, Sep. 2016.

[12] Y. Shen, M. Mardani, and G. B. Giannakis, “Online categorical subspace learning for sketching big data with misses,” *IEEE Trans. Signal Processing*, vol. 65, no. 15, pp. 4004–4018, August 2017.

[13] Y. Shen, T. Chen, and G. B. Giannakis, “Online ensemble multi-kernel learning adaptive to non-stationary and adversarial environments,” in *Proc. of Intl. Conf. on Artificial Intelligence and Statistics*, Lanzarote, Canary Islands, Apr. 2018.

[14] ——, “Online learning adaptive to dynamic and adversarial environments,” in *Intl. Workshop on Signal Processing Advances in Wireless Communications*, Kalamata, Greece, Jun. 2018.

[15] ——, “Random feature-based online multi-kernel learning in environments with unknown dynamics,” *Journal of Machine Learning Research*, vol. 20, no. 1, pp. 773–808, 2019.

[16] ——, “Online multi-kernel learning with orthogonal random features,” in *Proc. of Intl. Conf. on Acoust., Speech, and Signal Processing*, Calgary, AB, Canada, Apr. 2018.

[17] V. N. Ioannidis, P. A. Traganitis, Y. Shen., and G. B. Giannakis, “Kernel-based semi-supervised learning over multilayer graphs,” Kalamata, Greece, Jun. 2018.

[18] V. N. Ioannidis, Y. Shen., and G. B. Giannakis, “Semi-blind inference of topologies and signals over graphs,” in *Intl. Data Science Workshop*, Lausanne, Switzerland, Jun. 2018.

[19] V. N. Ioannidis, Y. Shen., and G. B. Giannakis, “Joint inference of topologies and signals over graphs,” submitted, 2018.

[20] Y. Shen, B. Baingana, and G. B. Giannakis, "Inferring directed network topologies via tensor factorization," in *Proc. of Asilomar Conf.*, Pacific Grove, CA, Nov. 2016.

[21] Y. Shen, X. Fu, G. B. Giannakis, and N. D. Sidiropoulos, "Directed network topology inference via sparse joint diagonalization," in *Proc. of Asilomar Conf.*, Pacific Grove, CA, Nov. 2017.

[22] Y. Shen, B. Baingana, and G. B. Giannakis, "Tracking dynamic piecewise-constant network topologies via adaptive tensor factorization," in *Proc. of Glob. Conf. Sig. and Info. Process.*, Washington, DC, Dec. 2016.

[23] ——, "Tensor decompositions for identifying directed graph topologies and tracking dynamic networks," *IEEE Trans. Signal Processing*, vol. 65, no. 14, pp. 3675–3687, July 2017.

[24] ——, "Nonlinear structural equation models for network topology inference," in *Proc. of Conf. on Info. Sciences and Systems*, Princeton, NJ, Mar. 2016.

[25] ——, "Topology inference of directed graphs using nonlinear structural vector autoregressive models," in *Proc. of Intl. Conf. on Acoustic Speech and Signal Process.*, New Orleans, USA, 2017, pp. 6513–6517.

[26] P. Traganitis, Y. Shen, and G. B. Giannakis, "Network topology inference via elastic net SEMs," in *EUSIPCO*, Kos Island, Greece, Aug. 2017.

[27] Y. Shen, B. Baingana, and G. B. Giannakis, "Nonlinear structural vector autoregressive models for inferring effective brain network connectivity," *IEEE Trans. on Signal Processing*, 2016 (submitted). [Online]. Available: https://arxiv.org/abs/1610.06551

[28] ——, "Kernel-based structural equation models for topology identification of directed networks," *IEEE Trans. Sig. Proc.*, vol. 65, no. 10, pp. 2503–2516, May 2017.

[29] J. Chen, G. Wang, **Y. Shen**, and G. B. Giannakis, "Canonical correlation analysis on graphs," in *Statistical Signal Processing Workshop*, Freiburg, Germany, Jun. 2018.

[30] ——, "Nonlinear canonical correlation analysis over graphs," *IEEE Transactions on Signal Processing*, vol. 66, no. 16, pp. 3398–4408, August 2018.

[31] Y. Shen, P. Traganitis, and G. B. Giannakis, “Nonlinear dimensionality reduction on graphs,” in *Proc. of CAMSAP*, Dutch Antilles, Dec. 2017.

[32] ——, “Graph-adaptive nonlinear dimensionality reduction,” *IEEE Trans. Signal Processing*, submitted April 2017.

[33] Y. Shen, G. B. Giannakis, and G. Leus, “Online graph-adaptive learning with scalability and privacy,” *IEEE Transactions on Signal Processing*, submitted August 2018, revised November 2018.

[34] M. Koyutürk and A. Grama, “PROXIMUS: A framework for analyzing very high dimensional discrete-attributed datasets,” in *Proc. ACM SIGKDD Intl. Conf. Knowledge Discovery and Data Mining*, Washington, US, Aug. 2003, pp. 147–156.

[35] B.-H. Shen, S. Ji, and J. Ye, “Mining discrete patterns via binary matrix factorization,” in *Proc. Intl. Conf. on Knowledge Discovery and Data Mining*, Paris, France, Jul. 2009, pp. 757–766.

[36] M. Koyuturk, A. Grama, and N. Ramakrishnan, “Compression, clustering, and pattern discovery in very high-dimensional discrete-attribute data sets,” *Trans. Knowledge and Data Engineering*, vol. 17, no. 4, pp. 447–461, Apr. 2005.

[37] H. Lu, J. Vaidya, V. Atluri, H. Shin, and L. Jiang, “Weighted rank-one binary matrix factorization.” in *Proc. of SIAM Intl. Conf. on Data Mining*, Mesa, USA, Apr. 2011, pp. 283–294.

[38] D. L. Rohde, “Methods for binary multidimensional scaling,” *Neural Computation*, vol. 14, no. 5, pp. 1195–1232, May 2002.

[39] Z. Huang, “A fast clustering algorithm to cluster very large categorical data sets in data mining.” in *Proc. Workshop on Research Issues on Data Mining and Knowledge Discovery*, Tucson, Arizona, May 1997, pp. 1–8.

[40] J. MacQueen, “Some methods for classification and analysis of multivariate observations,” in *Proc. Berkeley Symposium on Mathematical Statistics and Probability*, vol. 1, no. 14, Oakland, USA, Jan. 1967, pp. 281–297.

[41] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[42] J. Mairal, J. Bach, J. Ponce, and G. Sapiro, "Online learning for matrix factorization and sparse coding," *J. of Machine Learning Research*, vol. 11, pp. 19–60, Jan. 2010.

[43] J. Mairal, "Stochastic majorization-minimization algorithms for large-scale optimization," in *Advances in Neural Information Processing Systems*, 2013, pp. 2283–2291.

[44] M. Mardani, G. Mateos, and G. B. Giannakis, "Subspace learning and imputation for streaming big data matrices and tensors," *IEEE Trans. Sig. Proc.*, vol. 63, no. 10, pp. 2663–2677, May 2015.

[45] D. Collett, *Modelling survival data in medical research*. CRC press, 2015.

[46] D. Berberidis, V. Kekatos, and G. B. Giannakis, "Online censoring for large-scale regressions with application to streaming big data," *IEEE Trans. Sig. Proc.*, vol. 64, no. 15, pp. 3854–3867, Aug. 2015.

[47] T. Hastie, R. Tibshirani, and J. Friedman, *The Elements of Statistical Learning*. Springer, 2009.

[48] M. Fazel, "Matrix rank minimization with applications," Ph.D. dissertation, Stanford University, 2002.

[49] M. A. Davenport, Y. Plan, E. van den Berg, and M. Wootters, "1-bit matrix completion," *Information and Inference*, vol. 3, no. 3, pp. 189–223, Jul. 2014.

[50] M. Mardani, G. Mateos, and G. B. Giannakis, "Dynamic anomalography: Tracking network anomalies via sparsity and low rank," *IEEE J. Sel. Topics Sig. Proc.*, vol. 7, no. 1, pp. 50–66, Feb. 2013.

[51] Y. Shen, M. Mardani, and G. B. Giannakis, "Online sketching of big categorical data with absent features," in *Proc. of Conf. on Info. Sciences and Systems*, Baltimore, MD, Mar. 2015.

[52] S. P. Kasiviswanathan, H. Wang, A. Banerjee, and P. Melville, "Online ℓ_1-dictionary learning with application to novel document detection," in *Proc. of Neural Info. Proc. Systems*, Lake Tahoe, Dec. 2012, pp. 2258–2266.

[53] S. Shalev-Shwartz, “Online learning and online convex optimization,” *Found. and Trends in Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.

[54] B. Li, M. Ma, and G. B. Giannakis, “On the convergence of SARAH and beyond,” *arXiv preprint arXiv:1906.02351*, 2019.

[55] V. Solo and X. Kong, *Adaptive Signal Processing Algorithms: Stability and Performance*. Prentice Hall, 1995.

[56] R. Ayoub, “Euler and the zeta function,” *The American Mathematical Monthly*, vol. 81, no. 10, pp. 1067–1086, Dec. 1974.

[57] T. G. Dietterich, “An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization,” *Machine Learning*, pp. 1–22, 1999.

[58] B. N. Miller, I. Albert, S. K. Lam, J. Konstan, and J. Riedl, “Movielens unplugged: Experiences with an occasionally connected recommender system,” in *Proc. of Intl. Conf. on Intell. User Inter.*, Miami, FL, USA, Jan. 2003, pp. 263–266.

[59] B. Schölkopf and A. J. Smola, *Learning with Kernels*. Cambridge, MA: MIT Press, 2002.

[60] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, United Kingdom: Cambridge University Press, 2004.

[61] B. Dai, N. He, Y. Pan, B. Boots, and L. Song, “Learning from conditional distributions via dual embeddings,” in *Proc. Intl. Conf. on Artificial Intelligence and Statistics*, Fort Lauderdale, FL, Apr. 2017, pp. 1458–1467.

[62] A. Rakotomamonjy, F. R. Bach, S. Canu, and Y. Grandvalet, “SimpleMKL,” *J. Machine Learning Res.*, vol. 9, pp. 2491–2521, Nov. 2008.

[63] C. Cortes, M. Mohri, and A. Rostamizadeh, “ℓ_2-regularization for learning kernels,” in *Proc. Conf. on Uncertainty in Artificial Intelligence*, Montreal, Canada, Jun. 2009, pp. 109–116.

[64] M. Gönen and E. Alpaydın, "Multiple kernel learning algorithms," *J. Machine Learning Res.*, vol. 12, pp. 2211–2268, Jul. 2011.

[65] J. A. Bazerque and G. B. Giannakis, "Nonparametric basis pursuit via sparse kernel-based learning: A unifying view with advances in blind methods," *IEEE Sig. Process. Mag.*, vol. 30, pp. 112–125, Jul. 2013.

[66] J. Ma, L. K. Saul, S. Savage, and G. M. Voelker, "Identifying suspicious URLs: An application of large-scale online learning," in *Proc. Intl. Conf. Mach. Learn.*, Montreal, Canada, Jun. 2009, pp. 681–688.

[67] C. Richard, J. C. M. Bermudez, and P. Honeine, "Online prediction of time series data with kernels," *IEEE Trans. Sig. Proc.*, vol. 57, no. 3, pp. 1058–1067, Mar. 2009.

[68] J. Kivinen, A. J. Smola, and R. C. Williamson, "Online learning with kernels," *IEEE Trans. Sig. Proc.*, vol. 52, no. 8, pp. 2165–2176, Aug. 2004.

[69] S. C. Hoi, R. Jin, P. Zhao, and T. Yang, "Online multiple kernel classification," *Machine Learning*, vol. 90, no. 2, pp. 289–316, Feb. 2013.

[70] G. Wahba, *Spline Models for Observational Data.* Philadelphia, PA: Society for Industrial and Applied Mathematics, 1990.

[71] O. Besbes, Y. Gur, and A. Zeevi, "Non-stationary stochastic optimization," *Operations Research*, vol. 63, no. 5, pp. 1227–1244, Sep. 2015.

[72] C. K. Williams and M. Seeger, "Using the Nyström method to speed up kernel machines," in *Proc. Advances in Neural Info. Process. Syst.*, Vancouver, Canada, Dec. 2001, pp. 682–688.

[73] F. Sheikholeslami, D. Berberidis, and G. B. Giannakis, "Large-scale kernel-based feature extraction via budgeted nonlinear subspace tracking," *arXiv preprint:1601.07947*, Jul. 2016.

[74] C. Cortes, M. Mohri, and A. Talwalkar, "On the impact of kernel approximation on learning accuracy," in *Proc. Intl. Conf. on Artificial Intelligence and Statistics*, Sardinia, Italy, May 2010, pp. 113–120.

[75] A. Rahimi and B. Recht, “Random features for large-scale kernel machines,” in *Proc. Advances in Neural Info. Process. Syst.*, Vancouver, Canada, Dec. 2007, pp. 1177–1184.

[76] 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 *Proc. Advances in Neural Info. Process. Syst.*, Montreal, Canada, Dec. 2014, pp. 3041–3049.

[77] F. Yu, A. T. Suresh, K. Choromanski, D. Holtmann-Rice, and S. Kumar, “Orthogonal random features,” in *Proc. Advances in Neural Info. Process. Syst.*, Barcelona, Spain, Dec. 2016, pp. 1975–1983.

[78] G. R. Lanckriet, N. Cristianini, P. Bartlett, L. E. Ghaoui, and M. I. Jordan, “Learning the kernel matrix with semidefinite programming,” *J. Machine Learning Res.*, vol. 5, pp. 27–72, Jan. 2004.

[79] F. R. Bach, “Consistency of the group lasso and multiple kernel learning,” *J. Machine Learning Res.*, vol. 9, pp. 1179–1225, Jun. 2008.

[80] O. Dekel, S. Shalev-Shwartz, and Y. Singer, “The forgetron: A kernel-based perceptron on a budget,” *SIAM J. Computing*, vol. 37, no. 5, pp. 1342–1372, Jan. 2008.

[81] Z. Wang, K. Crammer, and S. Vucetic, “Breaking the curse of kernelization: Budgeted stochastic gradient descent for large-scale svm training,” *J. Machine Learning Res.*, vol. 13, pp. 3103–3131, Oct. 2012.

[82] R. Jin, S. C. Hoi, and T. Yang, “Online multiple kernel learning: Algorithms and mistake bounds,” in *Proc. Intl. Conf. on Algorithmic Learning Theory*, Canberra, Australia, Oct. 2010, pp. 390–404.

[83] D. Sahoo, S. C. Hoi, and P. Zhao, “Cost sensitive online multiple kernel classification,” in *Proc. Asian Conf. Machine Learning*, Hamilton, New Zealand, Nov. 2016, pp. 65–80.

[84] D. Sahoo, S. C. Hoi, and B. Li, “Online multiple kernel regression,” in *Intl. Conf. on Know. Disc. and Data Mining*, New York, Aug. 2014, pp. 293–302.

[85] J. Lu, D. Sahoo, P. Zhao, and S. C. Hoi, “Sparse passive-aggressive learning for bounded online kernel methods,” *ACM Trans. Intell. Syst. Tech.*, vol. 9, no. 4, p. 45, Feb. 2018.

[86] J. Lu, S. C. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu, "Large scale online kernel learning," *J. Machine Learning Res.*, vol. 17, no. 47, pp. 1–43, Apr. 2016.

[87] P. Bouboulis, S. Chouvardas, and S. Theodoridis, "Online distributed learning over networks in RKH spaces using random Fourier features," *IEEE Trans. Sig. Proc.*, to appear, 2018.

[88] Y. Ding, C. Liu, P. Zhao, and S. C. Hoi, "Large scale kernel methods for online auc maximization," in *Proc. IEEE Intl. Conf. Data Mining*, Nov. 2017, pp. 91–100.

[89] E. Hazan, "Introduction to online convex optimization," *Found. and Trends in Mach. Learn.*, vol. 2, no. 3-4, pp. 157–325, 2016.

[90] C. A. Micchelli and M. Pontil, "Learning the kernel function via regularization," *J. of Mach. Learn. Res.*, vol. 6, pp. 1099–1125, 2005.

[91] V. G. Vovk, "A game of prediction with expert advice," in *Proc. Annual Conf. Computational Learning Theory*, Santa Cruz, CA, Jul. 1995, pp. 51–60.

[92] N. Cesa-Bianchi and G. Lugosi, *Prediction, Learning, and Games*. Cambridge, United Kingdom: Cambridge University Press, 2006.

[93] A. Daniely, A. Gonen, and S. Shalev-Shwartz, "Strongly adaptive online learning." in *Proc. Intl. Conf. on Machine Learning*, Lille, France, Jun. 2015, pp. 1405–1411.

[94] A. Jadbabaie, A. Rakhlin, S. Shahrampour, and K. Sridharan, "Online optimization: Competing with dynamic comparators," in *Intl. Conf. Artificial Intell. and Stat.*, San Diego, CA, May 2015.

[95] M. Lichman, "UCI machine learning repository," 2013. [Online]. Available: http://archive.ics.uci.edu/ml

[96] F. Kawala, A. Douzal-Chouakria, E. Gaussier, and E. Dimert, "Prédictions d'activité dans les réseaux sociaux en ligne," in *4ième Conférence sur les Modèles et l'Analyse des Réseaux: Approches Mathématiques et Informatiques*, 2013.

[97] 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.

[98] 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, Feb. 2008.

[99] A. Rudi and L. Rosasco, “Generalization properties of learning with random features,” in *Proc. Advances in Neural Info. Process. Syst.*, Long Beach, CA, Dec. 2017, pp. 3218–3228.

[100] D. Bacciu, P. Barsocchi, S. Chessa, C. Gallicchio, and A. Micheli, “An experimental characterization of reservoir computing in ambient assisted living applications,” *Neural Computing and Applications*, vol. 24, no. 6, pp. 1451–1464, May 2014.

[101] J. Lines, A. Bagnall, P. Caiger-Smith, and S. Anderson, “Classification of household devices by electricity usage profiles,” in *Intl. Conf. on Intelligent Data Engineering and Automated Learning*, Norwich, United Kingdom, Sept. 2011, pp. 403–412.

[102] D. Anguita, A. Ghio, L. Oneto, X. Parra, and J. L. Reyes-Ortiz, “A public domain dataset for human activity recognition using smartphones.” in *Euro. Symp. on Artificial Neural Netw., Comp. Intell. and Mach. Learn.*, Bruges, Belgium, Apr. 2013.

[103] H. Luo and R. E. Schapire, “Achieving all with no parameters: Adanormalhedge,” in *Proc. Conf. on Learning Theory*, Lille, France, Jul. 2015, pp. 1286–1304.

[104] H. Luo, A. Agarwal, and J. Langford, “Efficient contextual bandits in non-stationary worlds,” *arXiv preprint:1708.01799*, Aug. 2017.

[105] D. Easley and J. Kleinberg, *Networks, Crowds, and Markets: Reasoning about a Highly Connected World.* NY, USA: Cambridge Press, 2010.

[106] E. M. Rogers, *Diffusion of Innovations.* Washington: Free Press, 1995.

[107] M. Rubinov and O. Sporns, “Complex network measures of brain connectivity: Uses and interpretations,” *NeuroImage*, vol. 52, no. 3, pp. 1059–1069, Sep. 2010.

[108] M. G. Rodriguez, D. Balduzzi, and B. Schölkopf, “Uncovering the temporal dynamics of diffusion networks,” in *Proc. Intl. Conf. Mach. Learn.*, Bellevue, WA, USA, Jul. 2011.

[109] S. Myers and J. Leskovec, “On the convexity of latent social network inference,” in *Proc. Neural Inf. Process. Syst. Conf.*, Vancouver, BC, Canada, Feb. 2010, pp. 1741–1749.

[110] M. Gomez Rodriguez, J. Leskovec, and A. Krause, “Inferring networks of diffusion and influence,” in *Proc. ACM SIGKDD Intl. Conf. on Know. Discov. and Data Mining*, Washington, DC, USA, July 2010, pp. 1019–1028.

[111] A. S. Goldberger, “Structural equation methods in the social sciences,” *Econometrica*, vol. 40, no. 6, pp. 979–1001, Nov. 1972.

[112] B. Baingana, G. Mateos, and G. B. Giannakis, “Proximal-gradient algorithms for tracking cascades over social networks,” *IEEE J. Sel. Topics Sig. Proc.*, vol. 8, no. 4, pp. 563–575, Aug. 2014.

[113] X. Cai, J. A. Bazerque, and G. B. Giannakis, “Inference of gene regulatory networks with sparse structural equation models exploiting genetic perturbations,” *PLoS Comp. Biol.*, vol. 9, p. e1003068, 2013.

[114] J. A. Bazerque, B. Baingana, and G. B. Giannakis, “Identifiability of sparse structural equation models for directed and cyclic networks,” in *Proc. of Glob. Conf. on Sig. and Info. Process.*, Austin, TX, Dec. 2013.

[115] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM Review*, vol. 51, no. 3, pp. 455–500, Aug. 2009.

[116] Y. Shen, B. Baingana, and G. B. Giannakis, “Inferring directed network topologies via tensor factorization,” in *Proc. of Asilomar Conf.*, Pacific Grove, CA, Nov. 2016.

[117] ——, “Tracking dynamic piecewise-constant network topologies via adaptive tensor factorization,” in *Proc. of Glob. Conf. Sig. and Info. Process.*, Washington, DC, Dec. 2016.

[118] R. A. Harshman, “Foundations of the PARAFAC procedure: Models and conditions for an explanatory multi-modal factor analysis,” *UCLA Working Papers in Phonetics*, vol. 16, pp. 1–84, 1970.

[119] N. D. Sidiropoulos, L. De Lathauwer, X. Fu, K. Huang, E. E. Papalexakis, and C. Faloutsos, “Tensor decomposition for signal processing and machine learning,” 2016. [Online]. Available: http://arxiv.org/abs/1607.01668

[120] N. D. Sidiropoulos, G. B. Giannakis, and R. Bro, “Blind PARAFAC receivers for DS-CDMA systems,” *IEEE Trans. Sig. Proc.*, vol. 48, no. 3, pp. 810–823, Mar. 2000.

[121] K.-K. Lee, W.-K. Ma, X. Fu, T.-H. Chan, and C.-Y. Chi, “A Khatri–Rao subspace approach to blind identification of mixtures of quasi-stationary sources,” *Signal Processing*, vol. 93, no. 12, pp. 3515–3527, Dec. 2013.

[122] D. Nion, K. N. Mokios, N. D. Sidiropoulos, and A. Potamianos, “Batch and adaptive PARAFAC-based blind separation of convolutive speech mixtures,” *IEEE Trans. Audio, Speech Lang. Process.*, vol. 18, no. 6, pp. 1193–1207, Aug. 2010.

[123] A. Anandkumar, R. Ge, D. Hsu, and S. M. Kakade, “A tensor approach to learning mixed membership community models.” *J. of Mach. Learn. Res.*, vol. 15, no. 1, pp. 2239–2312, June 2014.

[124] E. E. Papalexakis, L. Akoglu, and D. Ience, “Do more views of a graph help? Community detection and clustering in multi-graphs,” in *Intl. Conf. Info. Fusion*, Askeri Museum, Istanbul, Turkey, July 2013, pp. 899–905.

[125] D. Nion and N. D. Sidiropoulos, “Adaptive algorithms to track the PARAFAC decomposition of a third-order tensor,” *IEEE Trans. Sig. Proc.*, vol. 57, no. 6, pp. 2299–2310, Mar. 2009.

[126] M. Gomez Rodriguez, J. Leskovec, and B. Schölkopf, “Structure and dynamics of information pathways in online media,” in *Proc. ACM Intl. Conf. Web Search and Data Mining*, Rome, Italy, Dec. 2013, pp. 23–32.

[127] D. Angelosante and G. B. Giannakis, “Sparse graphical modeling of piecewise-stationary time series,” in *Proc. Int. Conf. Acoust. Speech Signal Process.*, Prague, Czech Republic, May 2011, pp. 1960–1963.

[128] M. Tahani, A. Hemmatyar, H. R. Rabiee, and M. Ramezani, “Inferring dynamic diffusion networks in online media,” *ACM Transactions on Knowledge Discovery from Data*, vol. 10, no. 4, p. 44, July 2016.

[129] D. T. Pham and J. F. Cardoso, “Blind separation of instantaneous mixtures of nonstationary sources,” *IEEE Trans. Sig. Proc.*, vol. 49, no. 9, pp. 1837–1848, May 2001.

[130] A. R. Benson, D. F. Gleich, and J. Leskovec, “Tensor spectral clustering for partitioning higher-order network structures,” in *Proc. SIAM Intl. Conf. on Data Mining*, Vancouver, Canada, Feb. 2015, pp. 118–126.

[131] F. Sheikholeslami, B. Baingana, G. B. Giannakis, and N. D. Sidiropoulos, “Egonet tensor decomposition for community identification,” in *Proc.of Glob. Conf. Sig. and Info. Process.*, Washington, DC, Dec. 2016.

[132] A. L. F. de Almeida, A. Y. Kibangou, S. Miron, and D. C. Araujo, “Joint data and connection topology recovery in collaborative wireless sensor networks,” in *Proc. Intl. Conf. on Acoustics, Speech, and Signal Process.*, Vancouver, Canada, May 2013.

[133] D. Kaplan, *Structural Equation Modeling: Foundations and Extensions*. Sage, 2009, vol. 2nd ed.

[134] J. B. Kruskal, “Three-way arrays: Rank and uniqueness of trilinear decompositions, with application to arithmetic complexity and statistics,” *Linear Algebra and its Applications*, vol. 18, no. 2, pp. 95–138, 1977.

[135] J. D. Carroll and J.-J. Chang, “Analysis of individual differences in multidimensional scaling via an n-way generalization of “Eckart-Young” decomposition,” *Psychometrika*, vol. 35, no. 3, pp. 283–319, 1970.

[136] A. Stegeman and N. D. Sidiropoulos, “On Kruskal’s uniqueness condition for the CAMDECOMP/PAEAFAC decomposition,” *Linear Algebra and its Applications*, vol. 420, no. 2, pp. 540–552, Jan. 2007.

[137] M. Sørensen and L. De Lathauwer, “New uniqueness conditions for the canonical polyadic decomposition of third-order tensors,” *SIAM Journal on Matrix Analysis and Applications*, vol. 36, no. 4, pp. 1381–1403, Oct. 2015.

[138] G. H. Golub and C. F. V. Loan, *Matrix Computations*. Johns Hopkins University Press, 1996.

[139] J. Leskovec, D. Chakrabarti, J. Kleinberg, C. Faloutsos, and Z. Ghahramani, “Kronecker graphs: An approach to modeling networks,” *J. Mach. Learn. Res.*, vol. 11, pp. 985–1042, Mar. 2010.

[140] N. Vervliet, O. Debals, L. Sorber, M. Van Barel, and L. De Lathauwer, "Tensorlab 3.0," Mar. 2016. [Online]. Available: http://www.tensorlab.net

[141] [Online]. Available: http://finance.yahoo.com/lookup?s=API

[142] J. S. Weszka, "A survey of threshold selection techniques," *Computer Graphics and Image Processing*, vol. 7, no. 2, pp. 259–265, Apr. 1978.

[143] R. Roche and S. Commins, *Pioneering Studies in Cognitive Neuroscience*. McGraw-Hill Education, 2009.

[144] K. J. Friston, C. D. Frith, and R. S. Frackowiak, "Time-dependent changes in effective connectivity measured with PET," *Human Brain Mapping*, vol. 1, no. 1, pp. 69–79, Jan. 1993.

[145] C. Büchel and K. J. Friston, "Modulation of connectivity in visual pathways by attention: Cortical interactions evaluated with structural equation modelling and fMRI." *Cerebral Cortex*, vol. 7, no. 8, pp. 768–778, Dec. 1997.

[146] R. Goebel, A. Roebroeck, D.-S. Kim, and E. Formisano, "Investigating directed cortical interactions in time-resolved fMRI data using vector autoregressive modeling and Granger causality mapping," *Magnetic Resonance Imaging*, vol. 21, no. 10, pp. 1251–1261, Dec. 2003.

[147] D. R. Gitelman, W. D. Penny, J. Ashburner, and K. J. Friston, "Modeling regional and psychophysiologic interactions in fMRI: The importance of hemodynamic deconvolution," *NeuroImage*, vol. 19, no. 1, pp. 200–207, May 2003.

[148] G. Deshpande, X. Hu, R. Stilla, and K. Sathian, "Effective connectivity during haptic perception: A study using Granger causality analysis of functional magnetic resonance imaging data," *NeuroImage*, vol. 40, no. 4, pp. 1807–1814, May 2008.

[149] K. J. Friston, "Functional and effective connectivity in neuroimaging: A synthesis," *Human Brain Map.*, vol. 2, no. 1-2, pp. 56–78, Jan. 1994.

[150] A. Roebroeck, E. Formisano, and R. Goebel, "Mapping directed influence over the brain using Granger causality and fMRI," *NeuroImage*, vol. 25, no. 1, pp. 230–242, Mar. 2005.

[151] A. Mclntosh and F. Gonzalez-Lima, “Structural equation modeling and its application to network analysis in functional brain imaging,” *Human Brain Mapping*, vol. 2, no. 1-2, pp. 2–22, Oct. 1994.

[152] K. J. Friston, L. Harrison, and W. Penny, “Dynamic causal modelling,” *NeuroImage*, vol. 19, no. 4, pp. 1273–1302, Aug. 2003.

[153] G. Chen, D. R. Glen, Z. S. Saad, J. P. Hamilton, M. E. Thomason, I. H. Gotlib, and R. W. Cox, “Vector autoregression, structural equation modeling, and their synthesis in neuroimaging data analysis,” *Computers in Biology and Medicine*, vol. 41, no. 12, pp. 1142–1155, Dec. 2011.

[154] K. G. Jöreskog, F. Yang, G. Marcoulides, and R. Schumacker, “Nonlinear structural equation models: The Kenny-Judd model with interaction effects,” *Advanced Structural Equation Modeling: Issues and Techniques*, pp. 57–88, 1996.

[155] M. Wall and Y. Amemiya, “Estimation for polynomial structural equation models,” *J. Amer. Stat. Assoc.*, vol. 95, pp. 929–940, 2000.

[156] X. Jiang, S. Mahadevan, and A. Urbina, “Bayesian nonlinear structural equation modeling for hierarchical validation of dynamical systems,” *Mech. Sys. and Sig. Proc.*, vol. 24, no. 4, pp. 957–975, Apr. 2010.

[157] S.-Y. Lee and X.-Y. Song, “Model comparison of nonlinear structural equation models with fixed covariates,” *Psychometrika*, vol. 68, no. 1, pp. 27–47, Mar. 2003.

[158] J. R. Harring, B. A. Weiss, and J.-C. Hsu, “A comparison of methods for estimating quadratic effects in nonlinear structural equation models.” *Psychological Methods*, vol. 17, no. 2, pp. 193–214, Jun. 2012.

[159] A. Kelava, B. Nagengast, and H. Brandt, “A nonlinear structural equation mixture modeling approach for nonnormally distributed latent predictor variables,” *Structural Equation Modeling: A Multidisciplinary Journal*, vol. 21, no. 3, pp. 468–481, Jun. 2014.

[160] S.-Y. Lee, X.-Y. Song, and J. C. Lee, “Maximum likelihood estimation of nonlinear structural equation models with ignorable missing data,” *J. of Educ. and Behavioral Stat.*, vol. 28, no. 2, pp. 111–134, 2003.

[161] D. Marinazzo, M. Pellicoro, and S. Stramaglia, "Kernel-Granger causality and the analysis of dynamical networks," *Physical Review E*, vol. 77, no. 5, p. 056215, May 2008.

[162] ——, "Kernel method for nonlinear Granger causality," *Physical Review Letters*, vol. 100, no. 14, p. 144103, Apr. 2008.

[163] X. Sun, "Assessing nonlinear Granger causality from multivariate time series," in *Proc. Eur. Conf. Mach. Learn. Knowl. Disc. Databases*, Antwerp, Belgium, Sep. 2008, pp. 440–455.

[164] N. Lim, F. d'Alché Buc, C. Auliac, and G. Michailidis, "Operator-valued kernel-based vector autoregressive models for network inference," *Machine Learning*, vol. 99, no. 3, pp. 489–513, Jun. 2015.

[165] G. Varoquaux, A. Gramfort, J.-B. Poline, and B. Thirion, "Brain covariance selection: better individual functional connectivity models using population prior," in *Advances in Neural Info. Process. Sys.*, Vancouver, Canada, 2010, pp. 2334–2342.

[166] P. A. Valdés-Sosa, J. M. Sánchez-Bornot, A. Lage-Castellanos, M. Vega-Hernández, J. Bosch-Bayard, L. Melie-García, and E. Canales-Rodríguez, "Estimating brain functional connectivity with sparse multivariate autoregression," *Philosophical Trans. of the Royal Soc. of London B: Bio. Sci.*, vol. 360, no. 1457, pp. 969–981, May 2005.

[167] S. Haufe, R. Tomioka, G. Nolte, K.-R. Müller, and M. Kawanabe, "Modeling sparse connectivity between underlying brain sources for eeg/meg," *Trans. on Bio. Engineer.*, vol. 57, no. 8, pp. 1954–1963, May 2010.

[168] S. Chen and P. Gopalakrishnan, "Speaker, environment and channel change detection and clustering via the Bayesian information criterion," in *Proc. DARPA Broadcast News Transcription and Understanding Workshop*, Virginia, USA, Aug. 1998, pp. 127–132.

[169] H. Bozdogan, "Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions," *Psychometrika*, vol. 52, no. 3, pp. 345–370, Sep. 1987.

[170] D. P. Bertsekas, *Nonlinear Programming*, 2nd ed. Athena-Scientific, 1999.

[171] I. D. Schizas, A. Ribeiro, and G. B. Giannakis, “Consensus in ad hoc WSNs with noisy links Part I: Distributed estimation of deterministic signals,” *IEEE Trans. Sig. Proc.*, vol. 56, pp. 350–364, Jan. 2008.

[172] M. A. Kramer, E. D. Kolaczyk, and H. E. Kirsch, “Emergent network topology at seizure onset in humans,” *Epilepsy Research*, vol. 79, no. 2, pp. 173–186, May 2008.

[173] I. Daubechies, M. Defrise, and C. De Mol, “An iterative thresholding algorithm for linear inverse problems with a sparsity constraint,” *Comm. Pure Appl. Math.*, vol. 57, no. 11, pp. 1413–1457, Aug. 2004.

[174] B. Baingana, G. Mateos, and G. Giannakis, “Dynamic structural equation models for tracking topologies of social networks,” in *Proc. of Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, Dec. 2013, pp. 292–295.

[175] R. I. Kondor and J. Lafferty, “Diffusion kernels on graphs and other discrete structures,” in *Proc. Intl. Conf. on Machine Learning*, Sydney, Australia, Jul. 2002, pp. 315–322.

[176] M. Belkin, P. Niyogi, and V. Sindhwani, “Manifold regularization: A geometric framework for learning from labeled and unlabeled examples,” *J. of Mach. Learn. Res.*, vol. 7, pp. 2399–2434, Nov. 2006.

[177] L. Wasserman and J. D. Lafferty, “Statistical analysis of semi-supervised regression,” in *Advances in Neural Information Processing Systems*, Vancouver, Canada, 2008, pp. 801–808.

[178] Q. Lu and L. Getoor, “Link-based classification,” in *Proc. of Intl. Conf. on Machine Learning*, Washington DC, USA, 2003, pp. 496–503.

[179] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, “Topology identification and learning over graphs: Accounting for nonlinearities and dynamics,” *Proc. of the IEEE*, vol. 106, no. 5, pp. 787–807, May 2018.

[180] O. Chapelle, B. Scholkopf, and A. Zien, “Semi-supervised learning,” *IEEE Trans. Neural Netw.*, vol. 20, no. 3, pp. 542–542, Mar. 2009.

[181] O. Chapelle, V. Vapnik, and J. Weston, "Transductive inference for estimating values of functions," in *Advances in Neural Information Processing Systems*, Denver, CO, USA, 1999, pp. 421–427.

[182] C. Cortes and M. Mohri, "On transductive regression," in *Advances in Neural Inf. Process. Syst.*, Vancouver, Canada, Dec. 2006, pp. 305–312.

[183] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis, "Adaptive diffusions for scalable learning over graphs," *arXiv preprint arXiv:1804.02081*, 2018.

[184] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation in graph structured data," in *Proc IEEE Intl. Conf. Acoust. Speech Signal Process.*, Vancouver, Canada, 2013, pp. 5445–5449.

[185] X. Wang, P. Liu, and Y. Gu, "Local-set-based graph signal reconstruction," *IEEE Trans. Signal Process.*, vol. 63, no. 9, pp. 2432–2444, May 2015.

[186] D. Romero, M. Ma, and G. B. Giannakis, "Kernel-based reconstruction of graph signals," *IEEE Trans. on Sig. Process.*, vol. 65, no. 3, pp. 764–778, Feb. 2017.

[187] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Transactions on Signal Processing*, vol. 64, no. 7, pp. 1832–1843, April 2016.

[188] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Sig. Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[189] A. J. Smola and R. I. Kondor, "Kernels and regularization on graphs," in *Learning Theory and Kernel Machines*. Springer, 2003, pp. 144–158.

[190] V. N. Ioannidis, D. Romero, and G. B. Giannakis, "Inference of spatio-temporal functions over graphs via multikernel kriged Kalman filtering," *IEEE Trans. Signal Process.*, vol. 66, no. 12, pp. 3228–3239, 2018.

[191] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus, "Adaptive graph signal processing: Algorithms and optimal sampling strategies," *IEEE Trans. Sig. Process.*, vol. 66, no. 13, pp. 3584 – 3598, July 2018.

[192] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *The American Statistician*, vol. 46, no. 3, pp. 175–185, Feb. 1992.

[193] J. Chen, H. Fang, and Y. Saad, "Fast approximate kNN graph construction for high dimensional data via recursive Lanczos bisection," *Journal of Machine Learning Research*, vol. 10, pp. 1989–2012, September 2009.

[194] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *Journal of Complex Networks*, vol. 2, no. 3, pp. 203–271, 2014.

[195] P. Traganitis, Y. Shen, and G. B. Giannakis, "Topology inference for multilayer networks," in *Intl. Workshop on Network Science for Comms.*, Atlanta, GA, May 2017.

[196] D. Zhou, S. A. Orshanskiy, H. Zha, and C. L. Giles, "Co-ranking authors and documents in a heterogeneous network," in *Proc. of Intl. Conf. on Data Mining*, Omaha NE, USA, October 2007, pp. 739–744.

[197] Y. Sun and J. Han, "Mining heterogeneous information networks: A structural analysis approach," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 20–28, 2013.

[198] P. Erdos and A. Rényi, "On random graphs I," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.

[199] "Meteorology and climatology meteoswiss." [Online]. Available: http://www.meteoswiss.admin.ch/home/climate/past/climate-normals/climate-diagrams-and-normal-values-per-station.html

[200] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Trans. on Sig. Process.*, vol. 64, no. 23, pp. 6160–6173, Dec. 2016.

[201] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. on Knowledge Disc. from Data*, vol. 1, no. 1, Mar. 2007.