Learning Dynamic Connectivities and Signal Recovery over Graphs

Mahmoud Ramezani-Mayiami

UiA University of Agder

Doctoral Dissertations at the University of Agder 363

Learning Dynamic Connectivities and Signal Recovery over Graphs

Mahmoud Ramezani-Mayiami

Learning Dynamic Connectivities and Signal Recovery over Graphs

Doctoral Dissertation for the Degree *Philosophiae Doctor (Ph.D.)* at
the Faculty of Engineering and Science, Specialization in
Information and Communication Technology

University of Agder
Faculty of Engineering and Science
2023

Doctoral Dissertations at the University of Agder 363
ISBN: 978-82-8427-072-2
ISSN: 1504-9272

Print: 07 Media
Kristiansand

Sammendrag

Den produseres stadig mer data fra den virkelige verden, inkludert kundeforbruksdata, sosiale nettverksaktiviteter, økonomiske data, temperaturdata fra forskjellige regioner og målinger av operatørpaneler. Dette resulterer i raskt voksende datamengder. Å samle, overføre, lagre, hente og behandle disse enorme datavolumene er utfordrende på grunn av behovet for høye beregningsressurser og datalagringskapasitet. Men den viktigste oppgaven, og grunnen til at dataene ble samlet inn i utgangspunktet, er dataanalyse: Finne korrelasjoner, mønstre og sammenhenger, aggregere til høyere nivåer, og til slutt trekke ut nyttig informasjon og kunnskap. Nylig har rammeverket Graph Signal Processing (GSP) forenklet analysen av store datavolumer ved bruk av grafteori, der grafhjørner representerer komponentene i datanettverk av interesse. Dermed er forskjellige applikasjoner involvert i grafer som fanger den underliggende topologien mellom forskjellige enheter i nettverket. De fleste forskningsprosjektene i dette rammeverket prøver å manipulere de "klassiske" signalbehandlingskonseptene og lage en "grafisk" versjon ved å migrere fra en enhet til et nettverket av enheter. Resultatene er lovende, men det er fortsatt noen plasser for dette forskningsrammeverket for mer realistiske scenarier og applikasjoner.

Et av de utfordrende områdene i rammeverket for signalbehandling er å tilpasse en algoritme til nye data. Disse nye dataene kan være tidsprøver over en gitt tid eller nylig oppdaterte sensorvariabler på grunn av endringer i interne nettverksforhold. Denne Ph.D. avhandlingen undersøker dette emnet i GSP-domenet og løser problemet med adaptiv grafsignalbehandling. Som et resultat kan vi bruke GSP for et bredere spekter av applikasjoner, for eksempel ikke-stasjonære prosesser. Avhandlingen bruker tre perspektiver: 1) graftopologilæringen under ulike forhold, 2) adaptiv grafsignalrepresentasjon og gjenoppretting når sanntids grafprosesser er gitt, og 3) grafsignalrepresentasjonen i et nytt domene ved å bruke transformativ læring. Å være tilpasningsdyktig til nye data krever at løsningen har noen retningslinjer angående støy og avvik. Vi bruker realistiske antakelser, f.eks. signalglattheten over graftopologien, sparsomhet av dataavvik, sparsomheten til grafen, og støymodellen. Som et biprodukt estimerer vi også selve grafsignalet fra de

støyende målingene. Med andre ord, for å representere og gjenopprette grafsignalet fra forskjellige grafprosesser, undersøker denne avhandlingen dataadaptive algoritmer for å re-estimere graftopologien når en ny observasjon er tilgjengelig. Disse tilnærmingene er i stand til å forringe målingene og gjenopprette grafsignalene i støyende omgivelser. Hovedtrekk til de foreslåtte metodene er den lave beregningskompleksiteten, noe som fører til mulighet for nettbasert implementering.

For å representere grafsignalene i et transformdomene fokuserer denne avhandlingen på faktoranalysemodellen for Gaussian Markov random field (GMRF) prosesser og ordboklæringen for den generelle signalmodellen. Faktoranalysen representerer grafsignalene på grunnlag av egenvektorene til Laplace grafen. Det betyr at vi kan koble grafsignalene direkte til den underliggende topologien. Ved å bruke ordboklæringskonseptene, transformeres graftopologien og grafsignalene til ordbokdomenet og gir en ny fordelaktig representasjon for spesifikke bruksområder, som temperaturflyt i ulike regioner.

Abstract

The ever increasing rate of acquisition of real-world data sets including customer consumption data, social networks activities, financial data, temperature data from different regions, and brain-computer interface measurements results in rapidly growing data volumes. Collecting, transmitting, storing, retrieving, and processing these huge data volumes are challenging because of the need for high computational resources and data storage capacity. But the most important task, and the reason why the data was collected in the first place, is data analysis: Finding correlations, patterns and connections, aggregating to higher levels, and finally extracting useful information and knowledge. Recently, the Graph Signal Processing (GSP) framework has simplified the analysis of large data volumes by the use of graph theory, where graph vertices represent the components of the data network of interest. Thus, different applications are involved with graphs capturing the underlying topology among different entities of the network. Most of the research projects in this framework try to manipulate the "classical" signal processing concepts and make a "graphical" version by migrating from one sole entity to the network of entities. The results were promising but there are still some spaces for this research framework for more real-world scenarios and applications.

One of the challenging areas in the signal processing framework is to adapt an algorithm to new upcoming data. This new data can be time samples that are provided over time or recently updated sensor variables due to some changes in internal network conditions. In this Ph.D. thesis, we investigate this topic in the GSP domain and tackle the problem of adaptive graph signal processing. As a result, we can apply the GSP for a broader range of applications, like non-stationary processes. To aim it, we approach these three perspectives: 1) the graph topology learning under different conditions, 2) the adaptive graph signal representation and recovery when real-time graph processes are given, and 3) the graph signal representation in a new domain by using the transform learning's concept. Being adaptive to new data requires the solution to have some policies regarding noise and outliers. We apply some real-world assumptions here, e.g. the signal smoothness over the graph topology, the outlier sparsity, the sparsity of the graph, and the model of the noise.

As a by-product, we also estimate the graph signal itself from the noisy measurements. In other words, to represent and recover the graph signal from different graph processes, this dissertation investigates some data-adaptive algorithms to re-estimate the graph topology when a new observation is provided. Moreover, these approaches are capable of denoising the measurements and recovering the graph signals in noisy environments. The main characteristic of the proposed methods is the low computational complexity, leading to online implementation possibility.

To represent the graph signals in a transform domain, this dissertation focuses on the factor analysis model for Gaussian Markov random field (GMRF) processes and the dictionary learning for the general signal model. The factor analysis represents the graph signals on the basis, provided by the eigenvectors of the graph Laplacian. In this respect, we can connect the graph signals to the underlying topology directly. By applying the dictionary learning concepts, the graph topology and graph signals are transformed to the dictionary domain and have a new representation which is beneficial for specific applications, like temperature flow in different regions.

Preface

This dissertation is a result of the research work carried out at the Department of Information and Communication Technology (ICT), University of Agder (UiA), in Grimstad, Norway, from January 2016 to October 2019. The research work presented in this dissertation has been funded by UiA.

Production note: LaTeX has been adopted as the tool for writing this dissertation. The mathematical calculations and simulation results are obtained by using MATLAB and PYTHON.

Acknowledgments

Hereby, I would like to take this opportunity to show my gratitude to my supervisors, colleagues, friends, and family who gave me supports through all these years.

I am grateful to my supervisors, Prof. Andreas Prinz and Prof. Karl Skretting for their constant supports and guidance during this journey toward Ph.D. I was lucky to be one of their students and learned a lot from them. The most important thing that I have learned is to be patient when you face difficult situations in your life. Also, I acknowledge the supervision of Prof. Baltasar Beferull-Lozano who has been my supervisor at the beginning of my Ph.D. study. I also want to thank Emma Elisabeth Horneman for being the Ph.D. advisor and for all of her supports. I would also like to thank my best friend, Dr. Mohammad Hajimirsadeghi for all of his help, discussions, and for providing all the joint research opportunities and collaborations with very well-known researchers and professors at Princeton University. Last but foremost, my deepest appreciation goes to my beloved wife, Mohadeseh, who has stood by me through all my travails, my absences, and my impatience. She gave me support and help from the beginning of this journey to the end and beyond.

Mahmoud Ramezani-Mayiami
August 2023
Oslo, Norway

Contents

List of Figures

List of Tables

List of Abbreviations

ADMM	Alternating Direction Method of Multipliers
BCD	Block Coordinate Descent
CGL	Combinatorial Graph Laplacian
CGP	Causal Graph Process
DC	Direct Current
EM	Expectation Maximization
ER	Erdős Rényi
FISTA	Fast Iterative Shrinkage Thresholding Algorithms
fMRI	functionall Magnetic Resonance Imaging
FT	Fourier Transform
GD	Gradient Descent
GFT	Graph Fourier Transform
GMRF	Gaussian Markov Random Field
GRLS	Graph Recursive Least Square
GSO	Graph Shift Operator
GSP	Graph Signal Processing
i.i.d.	independent and identically distributed
ISTA	Iterative Shrinkage Thresholding Algorithms
KNN	K-Nearest Neighbor
LSI	Linear Shift-Invariant
MAP	Maximum Aposteriori Probability
MAR	Multivariate Auto Regressive
MMSE	Minimum Mean Squared Error
MRI	Magnetic Resonance Imaging
MSE	Mean Squared Error
NMI	Normalized Mutual Information
NMSD	Normalized Mean Squared Deviation
NMSE	Normalized Mean Squared Error
PDF	Probability Density Function

PCA	Principal Component Analysis
PD	Positive Definite
PSD	Positive Semi-Definite
RBF	Radial Basis Function
RLS	Recursive Least Square
SNR	Signal to Noise Ratio
VAR	Vector Auto Regressive

Chapter 1

Introduction

The current huge amount of existing non-structured data generated and collected anywhere and anytime has raised challenging issues of data storage, statistical processing, information inference, and so on. The sources of this high volume of data may be, for instance, wireless sensor networks installed to control and monitor some real-world processes, including several sensor variables at different locations. This network of sensors is connected based on an underlying topology that can be modeled by a graph in an abstract form. In many applications, we have no a priori information about the underlying structure, but it can be extracted from data and then used for prediction, filtering, and data inpainting. A graph learning algorithm may be proposed based on the given data, a priori knowledge about the underlying process, and some assumptions such as graph sparsity and signal smoothness. The graph sparsity refers to the fact that the number of available edges in a real-world graph is much less than that of a complete graph. Signal smoothness means that the data changes smoothly from one vertex to its neighbors and/or also at each vertex across time. These assumptions are reasonable in real applications, e.g. spatial relation of temperature in close regions where each region is only affected by a small number of other regions. Here, in this Ph.D. thesis, we are interested in solving some inference problems in the above-mentioned domain, where we are given a set of data series. Then, our main desired goals are to estimate the network topology, apply noise removal on graph signals, represent the graph signals in a transform domain by using the dictionary learning framework.

1.1 Background and Motivation

A signal is a function that ”conveys information about the behavior or attributes of some phenomena”. In the physical world, any quantity exhibiting variation in the time or space is potentially a signal, e.g. audio, video, speech, image, communication, geophysical, sonar, radar, medical and musical signals. The ”Signal Processing” framework concerns the analysis, synthesis, and modification of signals. ”Statistical Signal Processing” is an approach to signal processing that treats signals as stochastic processes, utilizing their statistical properties to perform signal processing tasks.

In this Ph.D. thesis, we investigate new statistical signal processing techniques for some inference tasks on dynamic graph processes. By inference, we mean more information extraction from the signal at hand. Examples of such inference tasks are signal reconstruction, filtering, and estimation of the underlying topology. A set of measurements of these signals are collected by a set of entities from real-world networks, such as a temperature sensor network, EEG sensors located on the head of a person, or social networks. In other words, the works in this area of research focus on multivariate signals, and the recently proposed framework of ”Graph Signal Processing” discusses many topics and problems around it. Here, we investigate some solutions for the problem of adaptivity for inference algorithms in the GSP framework. In other words, the main concern of this Ph.D. thesis is to approach adaptive graph signal processing.

Why should an algorithm be adaptive? When dealing with sequential data, time series data, or when the data comes from a non-stationary process, the currently proposed algorithm on the previous batch of the data is not working as it should be. The problem is mainly due to the changing nature of the data. In this situation, one solution is to adapt the proposed algorithm in some ways to the underlying data. This is a well-defined framework for many signal processing problems, like adaptive filters. In the GSP framework, this has not been discussed enough and needs more investigation. In this Ph.D. thesis, we try to tackle this problem for different scenarios and investigate solutions that help to implement GSP algorithms for a broader range of network data applications. To aim it, the following tasks should be done:

- Extending adaptive filter concept for graph signals,
- Applying dictionary learning methods to sparsely code the recently received graph signals (A sparse signal is a signal that has many zeros among its entries),

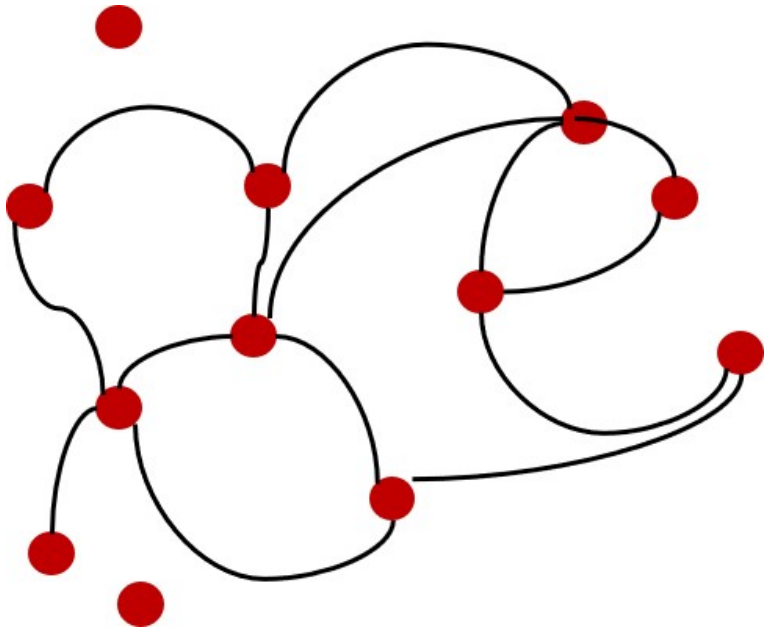


Figure 1.1: An example of the graph structure: red circles show the vertices and black lines are the edges.

- Using transform learning concepts to map the underlying structure to a new domain that can capture dependencies among graph signal coefficients in the transform domain,
- Learning the underlying graph structure in real-world scenarios and for different online applications.

1.1.1 Graph Signal Processing Framework

First of all, what is a graph? A graph can be defined simply as a set of ”entities” connected based on their ”relations”. In an abstract form, we show an entity by a ”vertex” and the relation between two entities by an ”edge” (Figure 1.1).

The emerging field of graph signal processing (GSP) [3, 8] provides a valuable tool for analysis of a large amount of data by leveraging a graph structure where each vertex represents the time series associated with a certain node variable, and where the edges capture the space-time dependencies. An edge weight usually represents the similarity between two end vertices.

The connectivities and edge weights can be dictated by the physics of the problem or inferred from the given set of data. For example in some geometrical applications, the weights are inversely proportional to the physical distance between sensor nodes in the network. By taking a snapshot of the values over such sensor nodes at a specific time instant, we collect different nodes' values in a vector, called a ”graph

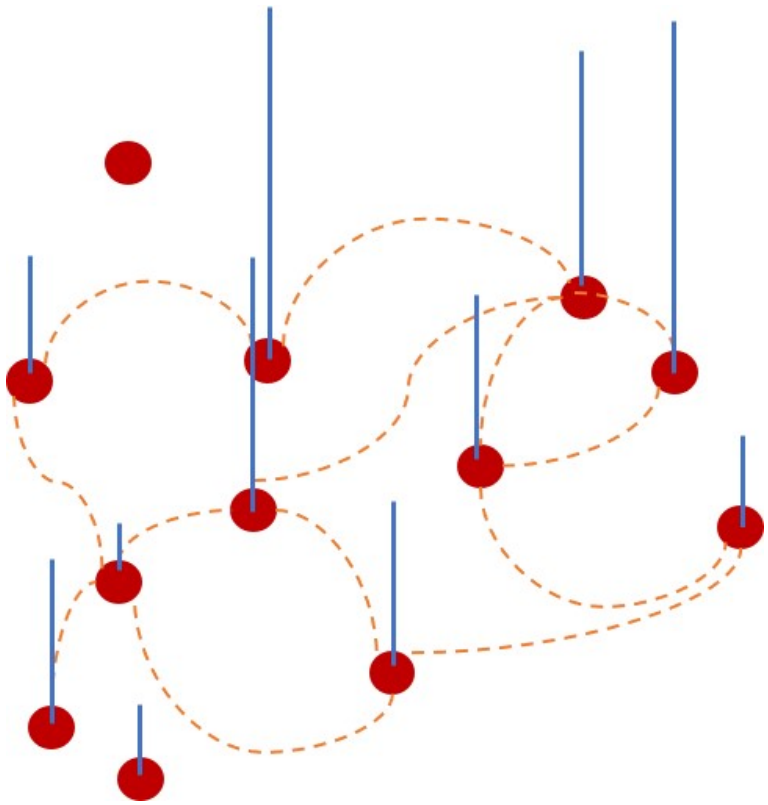


Figure 1.2: An example of a graph structure and a graph signal. The height of each blue bar represents the signal value at the vertex.

signal". In Figure (1.2), the red circle shows the vertex (or the sensor node), and the dashed line shows the underlying graph structure, and the height of each blue line represents the value of signal over the corresponding vertex (generally, these values can be positive or negative). By stacking all of these values in a vector, we have a graph signal. In this respect, a graph signal with N vertices can be shown by a vector in $\mathbb{R}^N$ which is the same way of representing a classical discrete-time signal with N samples. The main difference between these two is that a graph signal also conveys information of dependencies arising from the irregular data domain.

There are many examples of graph signals in real-world engineering problems. By using magnetic resonance imaging (MRI) or other brain image sampling methods, we infer the connectivity of the cerebral cortex and represent it by a weight adjacency matrix [9]. Another example is the automatic text classification [10], where a weighted graph is applied for statistical learning purposes. Graph signal filtering, signal denoising, and graph signal compression and representation are some of the other tasks in these graph signal processing applications.

Some of the general concerns of GSP include, but are not limited to

- The data processing in an irregular domain like an arbitrary graph,

- Extracting information from high dimensional multivariate signal to store, communicate, and analyze the data,
- Adapting the existing algorithm in classical signal processing framework for GSP, like graph Fourier transform (GFT),
- Learning the underlying topology from the given data set,
- Spectral analysis of high dimensional signals by leveraging the underlying graph structure.

The research works in this thesis mainly deal with two subjects of GSP; graph topology learning and graph signal representation.

1.1.2 Graph Topology Learning

The graph topology simply means how the vertices are connected. These connections are represented by a set of edges. In many applications, we know the graph topology in advance, e.g. classification of online blogs, damage identification in the bridge, structural health monitoring [11], signal inpainting on graphs via total variation minimization [12], sampling theory for graph signals [13] , and so on [14–26]. However, in some applications, we do not know the underlying structure and thus it is required to estimate it via the given measurements. Finding the flow of temperature in a thermal sensor network is one such application [27, 28]. Thus, we are interested in learning the underlying topology from the data at hand.

On one hand, some research works concentrated on directed topology estimation [27, 29–37], where specific process models have been assumed. On the other hand, many works have investigated undirected structures, such as [38–51, 51–53]. Following this research problem, in this Ph.D. thesis, we discuss the graph learning procedures and propose new methods for different scenarios and applications. We cover both the directed and undirected topology inference problems.

1.1.3 Signal Representation

One of the central topics in the classical signal processing framework is to represent the existing signal in another domain. A simple and widely used example of this domain is the Fourier transform (FT). In other words, we investigate a new domain for the signal to show it efficiently in some perspectives. A monotone sinusoidal signal is represented in the frequency domain by a pair of coefficients which is a sparse representation and suits compression applications.

Sometimes in the GSP framework, graph signals are represented in a new domain that fits the application. For example, [54] investigated a set of dictionary atoms to represent the graph signal with sparse coefficients. In this thesis, we try to represent the graph signals by different approaches, e.g. factor analysis model, dictionary learning, and auto-regressive processes generation. Moreover, the signal representation concept is used to recover the original signal from noisy measurements.

1.2 Research Questions

The objective of this research is to solve the following problem:

How can we infer the underlying graph topology from the given data, update it with the upcoming data, and estimate the true signals from noisy measurements?

To approach this question, we try to illuminate and hopefully answer the following sub-questions:

- **Question 1 (Q1)** Is it possible to extend the conventional adaptive filters for graph signals? If yes, how?

  Answering this question helps to provide adaptive connectivity between graph signals and the underlying structure and thus a graph topology can be learned out of it.

- **Question 2 (Q2)** How can we approach online graph topology inference for specific processes, e.g. Gaussian processes, Multivariate Auto-Regressive Processes (MAR), and Causal Graph Processes (CGP)?

  This question approaches the main question of the thesis by modeling the graph signals based on their generative processes. Therefore, the given multivariate data is modeled based on a known process and by finding the corresponding parameters, we can estimate the underlying structure.

- **Question 3 (Q3)** How to extend the concept of dictionary learning for the GSP framework?

  This is an instance of the general transform learning framework. Since we are interested in representing the graph signal in another domain, this question is important. Here, by representing the graph signals in the dictionary domain, we try to connect the topology inference problem to the dictionary atoms where the graph signals can be efficiently represented.

- **Question 4 (Q4)** How to relate the concept of atom coherence in dictionary learning with the underlying structure of the data?

  This question is close to Q3 and it is important once we tackle Q3. Both of Q3 and Q4 helps us finding an optimum topology which can be underlying the input graph signals represented on a set of approximately incoherent atoms.

- **Question 5 (Q5)** What kind of multivariate signal processes model can be connected to the underlying topology?

  In response to this question, we look after some processes that can provide a fit to the input data. Since the data is usually given by time sequentially, the investigated model adapts to the upcoming data and hence an adaptive topology can also be inferred as a by-product.

- **Question 6 (Q6)** How do we deal with noise which exists in graph signal measurements?

  This question concerns the graph signal recovery from contaminated measurements by noise. In the real-world data set, the data is not usually clean and hence we have to provide a solution for our main problem in the presence of noise.

- **Question 7 (Q7)** When the measurements are contaminated with a highly powerful sparse outlier, how can we modify the topology learning problem formulation?

  The answer to this question helps the solution of Q6, while here we should consider outlier instead of noise. Sometimes, both Q6 and Q7 are important at the same time and should be tackled simultaneously.

- **Question 8 (Q8)** To learn the topology and recovery of the graph signal, what kind of optimization problems may be formulated and how are the solution methods?

  This question is necessary to be asked when dealing with optimizing variables. For example, in this thesis's main question, we want to find the optimum graph Laplacian matrix and graph signals.

- **Question 9 (Q9)** How to implement a fast algorithm for graph topology learning and signal recovery for different kinds of multivariate data processes?

  Since we work with the big data, due to a network of several nodes which have lots of data samples over time, there is a concern about the complexity

Table 1.1: An overview of the identified research questions in different chapters.

	Ch. 3	Ch. 4	Ch. 5	Ch. 6
Q1				×
Q2				×
Q3			×	
Q4			×	
Q5	×	×	×	×
Q6	×	×	×	×
Q7		×		
Q8	×	×	×	×
Q9	×			×

of the algorithms when running in batch mode. Thus, when investigating the main question of this thesis, we also try to reduce the complexity.

Table 1.1 indicates the venues in this dissertation where the above-mentioned research questions are discussed and solutions proposed.

1.3 Research Directions

To answer the above question, the following directions should be followed in the research

- *Laplacian/adjacency matrix learning*: Based on the given processes and knowledge about the physics of the system, we propose methods for graph topology inference. The topology can be directed or undirected, depending on the application, for example, a directed graph for temperature sensor network or an undirected one that suits the stock market data,

- *Online graph topology inference*: By applying the traditional concept of adaptive filters to the graph signals, some methods will be proposed for online topology inference when new observation exists,

- *Graph signal denoising*: We also propose some methods to remove the noise from the given measurements over a graph.

- *Dictionary learning for graph signals*: We transfer signal and graph into another domain and define dictionary atoms, atom coherence, and sparse coefficients over a graph,

Considering the main problem discussed in this thesis which is to provide adaptive GSP algorithms for graph topology inference and signal recovery, it is required to include a research direction for topology inference offline and online which is followed by the first and second directions, respectively. The third item directs us to recover the signal out of noisy measurements or observations contaminated by outliers. The research done in the direction of dictionary learning for graph signals helps us representing the graph signal in another domain and is helpful in signal representation efficiently. In other words, these four items cover the entire goals of the thesis where we concentrate on the problem of adaptive graphs topology inference and signal representation.

1.4 Research Methodology

1.4.1 Literature Review

A solid literature review is the foundation of research and can help to uncover unsolved problems, important directions as well as problem-solving concepts. For this interdisciplinary work, the literature has to cover many different subjects. The research topic is directly at the intersection of graph theory and signal processing and hence the backgrounds of these two frameworks are required to be reviewed in the existing books and other teaching materials. Moreover, literature in the direction of graph signal processing has to be covered, including but not limited to the graph topology inference as well as the graph signal representation. The main sources for this literature are the recently published papers and keynote lectures of the signal processing society conferences. To be more specific, the following subject's literature are reviewed

- graph theory and its applications,
- directed and undirected topology learning,
- signal representation for multivariate signals,
- sparse signal processing,
- dictionary learning, sparse coding, atoms coherence and graph regularized dictionary learning,
- comprehensive studies of different processes for signal generations such as Gaussian processes, Causal Graph Processes, and Vector Auto-Regressive processes,

- adaptive filters, including recursive least square filter and Kalman filter,

1.4.2 Mathematical Analysis

The main line of research deals with graph theory and its application to network data analysis. The graph theory requires linear algebra, discrete mathematics, and solving optimization problems. Thus, one stage in every research direction of this thesis is to involve mathematical formulation and quantitative analysis. Besides, the signal processing framework needs some mathematical tools such as transforms and representation on some bases, adaptive filtering, and signal/noise modeling.

1.4.3 Design Algorithms

Solving the mathematical problem requires a practical implementation to approach the solution. Here, we have to propose some algorithms to implement the proposed solution. Besides the batch mode of algorithms, we try to make them adaptive to the upcoming data, i.e. we assume that the underlying graph structure is already estimated, and a new set of observations arrives and then we want to update our previous graph estimate.

1.4.4 Experiments

At the last stage, we examine the proposed algorithms on some data sets. Besides, we investigate the appropriate performance measures to evaluate the results.

1.4.4.1 Simulations with Synthetic Data

In the simulation, first, we want to apply the algorithms on some synthetic data sets which are generated based on the assumed process models.

1.4.4.2 Simulations with Real-World Data

To evaluate the proposed methods in practical conditions, we collect some real-world data sets from different sources. These data sets help us realizing which algorithm has a better performance for a specific signal process and application.

1.5 Research Contributions

The contributions of this dissertation are within the broad graph signal processing (GSP) field: Topology learning, graph signal recovery, and multivariate signal representation over the underlying structure. Even there are many methods in the literature that focus on these lines of research, in our proposed approaches, we consider different scenarios which are applicable in more realistic situations. The general contributions are outlined as follows.

In this dissertation, several methods are proposed for learning topology represented as a graph structure. In general, we can categorize all these learning methods as directed and undirected topology learning. The directed topology includes the topology of graphs inferring from CGP and MAR processes and undirected graphs can be extracted from Gaussian Markov Random Field (GMRF) processes. The CGP model captures the cause-effect relationships between the entities and by following this causality, we keep tracking the graph topology. Our main contribution for this scenario is to make the topology learning in an online fashion. We propose an RLS filter to process the received data sequentially. When the given data is from GMRF processes, we try to connect the data to the underlying structure, or in other words, the data input matrix is written as a function of the graph topology. Using the data fidelity term as well as signal smoothness property on the underlying topology and the edge sparsity, an optimization problem is formulated. Depending on the application, some constraints are applied. After solving the corresponding problem, an appropriate algorithm is suggested and efficiently implemented. The other contribution is applying the dictionary learning definition and atom coherence concepts to the GSP framework. In this respect, we formulate a problem to represent the graph signals in a transform domain. Some real-world assumptions are considered, e.g. reducing the average atom coherence, signal smoothness, and minimizing the data representation error.

Regarding the signal representation perspective, the function explained above, plays the main role. By using some statistical modeling tools, we propose a relevant function to connect the data to the underlying structure. A factor analysis model is used to represent GMRF processes with respect to the graph topology. For a directed topology, the CGP and MAR processes models connect each graph signal to the filtered version of others, where the filter is a function of the underlying topology. Finally, for all of the proposed methods, it is assumed that the signal measurements are contaminated by noise and thus it is more applicable in real engineering network problems.

1.6 Dissertation Outline

In general terms, this dissertation deals with the learning connectivities and graph signal recovery in widely used synthetic data models and real-world scenarios. These scenarios include temperature propagation in a geographical area and stock market data analysis.

Chapter 2 introduces the graph signal processing framework and its widely used concepts. The theoretical definitions from graph theory and the connection to classical signal processing are reviewed. Chapter 3 focuses on the GMRF process model which is highly applicable in undirected topology inference methods. The proposed method in this chapter uses the Bayesian inference concepts to filter signals and estimate the topology with higher performance with respect to the existing methods. Also, a fast algorithm is proposed to implement the algorithm. In Chapter 4, it is assumed that the graph measurements are contaminated by some outliers. A method is proposed to alternatively remove the outliers, learn the graph topology, and denoise the graph measurements. In Chapter 5, the dictionary learning concepts are adapted for the GSP framework. We represent graph signals with multivariate coefficients which are sparse and the transformed signals in the dictionary domain are smooth on the underlying graph. In Chapter 6, we consider some scenarios in which the data is received sequentially and hence propose a method based on adaptive filters which are relatively fast compared to the batch algorithms. Chapter 7 concludes the dissertation and points out a few potential extensions of the proposed schemes and models.

1.7 Historical Notes

The city of Königsberg was located on both sides of the Pregel River and included two islands connected to each other by seven bridges. The problem has been to design a walk through the city that would cross each of the bridges once and only once (Figure 1.3). There were two implicit conditions; it is not possible to reach an island without crossing a bridge, and both sides of a bridge must be met. The problem with these two conditions is called "The Seven Bridges of Königsberg" and it is a historically notable problem in mathematics. Leonhard Euler in 1736 proved that this problem has no solution and he paved the foundations of graph theory [55]. Euler's formula connects the number of edges, vertices, and faces of a convex polyhedron, and then it has been studied and manipulated by Cauchy and L'Huilier and formed the branch of mathematics known as topology.

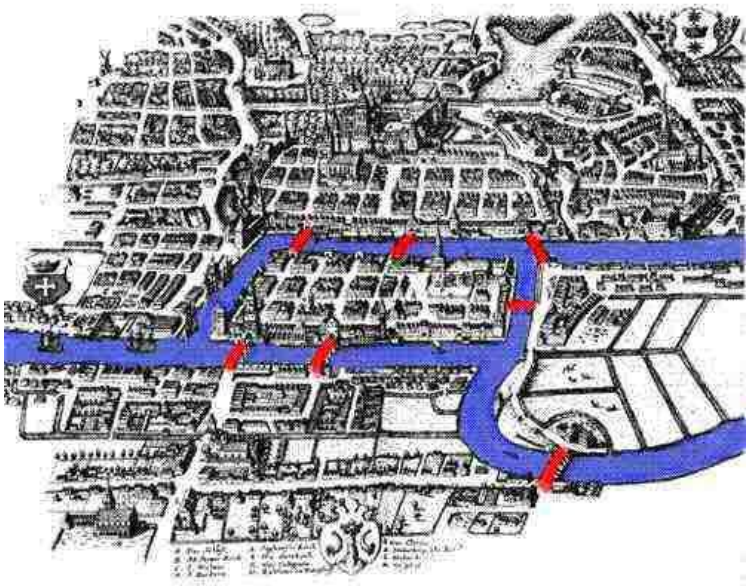


Figure 1.3: The seven bridges of the Königsberg problem [1].

However, later on, James Joseph Sylvester coined the term ”graph” formally and in a scientific manner [56]. *”... I give a rule for the geometrical multiplication of graphs, i.e. for constructing a graph to the product of in- or co-variants whose separate graphs are given...”*. This is one of the first sentences that Sylvester has said in a paper published in Nature in 1878.

Chapter 2

Graph Signal Processing

The graphs are utilized to model several types of relationships among entities in real-world applications. Some examples include, but are not limited to, information systems, customer consumption data in utility companies, biological and physical systems, different social networks, gene networks, financial data, and regional temperature data. In these examples, the system of interest has some entities or objects and there exist connections or relationships among them. From an abstract mathematical perspective, these entities are represented by "graph nodes" and the connections may be modeled by "graph edges". In this chapter, we connect the concept of multivariate signal processing to the graph theory and introduce the framework of graph signal processing. Different types of graphs along with some graph representation matrices are introduced. Some definitions and theorems from graph theory that apply to the rest of this dissertation will also be reviewed.

The emerging field of Graph Signal Processing (GSP) [3,8] simplifies the analysis of large data volumes by applying the graph theory. In applications such as social networks, energy consumption user data, transportation networks, and neuronal networks, high dimensional data can be considered as values over nodes in weighted graphs. The framework of signal processing on graphs merges algebraic and spectral graph-theoretical perspectives with computational harmonic analysis to process these multivariate signals. In this chapter, we outline some aspects of graph theory which is necessary to understand network topology and signal processing over graphs. Two important concepts of GSP will be introduced and explained: "graph", and the paradigm of signal over graph or "graph signal". In the rest, some mathematical definitions, properties, and theorems are presented which are important for

the main technical discussions and ideas of the next sections.

Throughout this dissertation, lowercase normal (e.g., a and θ), lowercase bold (e.g., $\mathbf{a}$ and $\boldsymbol{\theta}$) and uppercase bold (e.g., $\mathbf{A}$ and $\boldsymbol{\Theta}$) letters denote scalars, vectors, and matrices, respectively. The rest of the notation is presented in Table 2.1.

Table 2.1: Table of Notations

$\mathcal{G}$	graph
$\mathbf{W} \mid \mathbf{L}$	weight matrix| Laplacian matrix
$\mathbf{D} \mid \mathbf{A}$	degree matrix|adjacency matrix
$\mathcal{V} \mid \mathcal{E}$	vertex set|edge set
N	number of vertices or number of nodes in the graph
$\mathbf{0}_N \mid \mathbf{1}_N$	column vector of zeros with size N |column vector of ones with size N
$\boldsymbol{\Lambda} \mid \boldsymbol{\chi}$	eigenvalue matrix of $\mathbf{L}$ |eigenvector matrix of $\mathbf{L}$
$\boldsymbol{\Theta}^{-1} \mid \boldsymbol{\Theta}^{\dagger} \mid \boldsymbol{\Theta}^{T}$	inverse of $\boldsymbol{\Theta}$|psudo-inverse of $\boldsymbol{\Theta}$ | transpose of $\boldsymbol{\Theta}$
$\det(\boldsymbol{\Theta})\mid \lvert\boldsymbol{\Theta}\rvert$	determinant of $\boldsymbol{\Theta}$|psudo-determinant of $\boldsymbol{\Theta}$
$\theta_{ij} \mid \theta_i$	entry of $\boldsymbol{\Theta}$ at the ith row and the jth column| the i'th element of $\boldsymbol{\theta}$
$\boldsymbol{\Theta}^{(m)} \mid \boldsymbol{\Theta}^{m}$	the m'th matrix | the matrix $\boldsymbol{\Theta}$ to the power of m
$\boldsymbol{\Theta}_{i:} \mid \boldsymbol{\Theta}_{:j}$	i'th row of $\boldsymbol{\Theta}$ | j'th column of $\boldsymbol{\Theta}$
$\mathcal{S}^N \mid \mathcal{S}_+^N$	the set of symmetric | positive semi-definite matrices
Tr | logdet($\boldsymbol{\Theta}$)	trace operator|natural logarithm of det($\boldsymbol{\Theta}$)
$\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$	zero mean multivariate Gaussian with covariance $\boldsymbol{\Sigma}$
$\lVert\boldsymbol{\Theta}\rVert_F^2$	sum of squared values of all elements
$\lvert\theta_{ij}\rvert \mid \lVert\boldsymbol{\Theta}\rVert_1$	the absolute value of θ_{ij} | sum of absolute values of all elements
$\lVert\boldsymbol{\Theta}\rVert_0$	counts the number of non-zero entries
diag($\boldsymbol{\theta}$)	diagonal matrix formed by elements of $\boldsymbol{\theta}$
diag($\boldsymbol{\Theta}$)	vector of the diagonal elements of the input matrix
$\otimes \mid \odot$	Kronecker product|Hadamard (element-wise) product
$\mathbb{E}$	expectation operator
$p(\mathbf{a} \mid \mathbf{b})$	conditional probability density function of $\mathbf{a}$ given $\mathbf{b}$
$\hat{\boldsymbol{\theta}} \mid \overline{\boldsymbol{\theta}}$	estimate of $\boldsymbol{\theta}$ | mean of $\boldsymbol{\theta}$
sign($\boldsymbol{\theta}$)	sign of each element of $\boldsymbol{\theta}$
$\nabla \mid \nabla^2$	difference or gradient operator|hessian operator
$\mathbf{prox}_f$	proximal operator of function f
$\mathcal{L}$	Lagrangian function

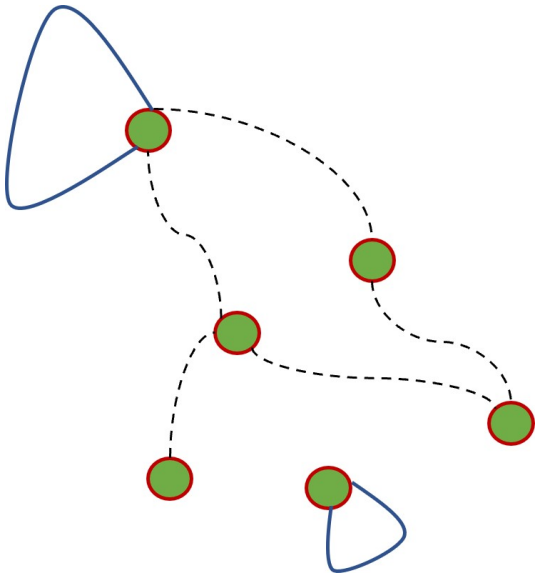


Figure 2.1: A graph with 6 vertices, 5 edges, and two self-loops.

2.1 Basics of the Graph Theory

2.1.1 Definition of Graph

The graph is a structure used for modeling the pairwise relationships among different entities. Therefore, the "entities" and "pairwise connections" play the most important roles in graph definitions. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a graph with vertices $v_i \in \mathcal{V}$ and the edge set $\mathcal{E}$, where each edge $(v_i, v_j), 1 \leq i, j \leq N$ shows a connection between two nodes. Sometimes, a vertex is also called a node or a point and an edge is called a link or a line. The "order" of a graph is $|\mathcal{V}|$, i.e. the number of vertices. The "size" of a graph is $|\mathcal{E}|$. If we consider two distinct vertices $v_i, v_j \in \mathcal{V}$, then the edge set can mathematically be represented as follows

$$\mathcal{E} = \{\{v_i, v_j\} \mid (v_i, v_j) \in \mathcal{V}^2\} \tag{2.1}$$

where the vertices v_i and v_j are also called the end points. If an edge connects a node to itself, that edge is called a loop or self-loop. Figure 2.1 shows a graph with six nodes, five edges, and two self-loops.

2.1.2 Graph Types

There are many types of a graph and in what follows, we only review some of them which are useful for the next chapters. The interested reader can refer to [57,58] for more details.

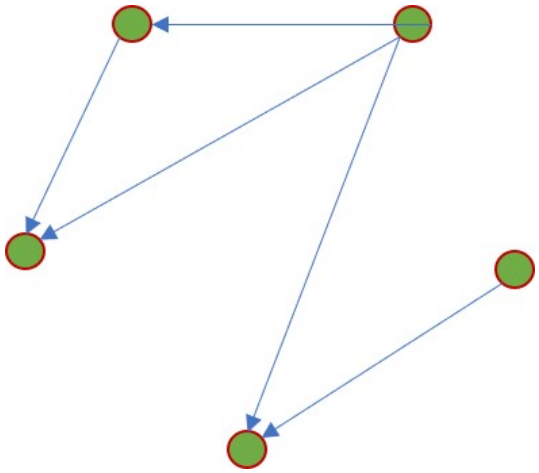


Figure 2.2: A directed graph

2.1.2.1 Directed graph

In this type of graph, each edge is directed from one vertex to another. This graph is also called "digraph" (Figure 2.2). A simple example of this type of graph is the graph of connections in the Twitter network. The person v_i may follow the person v_j but the opposite is not necessarily true.

The set of edges are sometimes called "directed edges", "arrows", "directed links", "arcs", and "directed lines" and they are ordered pairs of vertices. In a sample edge such as (v_i, v_j), directed from v_i to v_j, both v_i and v_j are called the endpoints where v_i is the tail and v_j is the head. Following this example, the inverted edge is called by (v_j, v_i). Directed graphs are very useful to model the cause effect relationships among entities.

2.1.2.2 Undirected graph

If all edges are bi-directional, the graph is called undirected (Figure 2.3). The edge is said to join v_i and v_j or to be incident on v_i and on v_j. The endpoints v_i and v_j are said to be adjacent to each other, which is denoted as $v_i \sim v_j$. A simple example of this type of graph is the "friendship" in the Facebook network. If person v_i is a friend of person v_j, the opposite is also held.

2.1.2.3 Weighted graph

In this type of graph, each edge $(v_i, v_j), 1 \leq i, j \leq N$ carries a weight. This weight can be zero, i.e. no edge, or any number that represents the strength of the connection between two nodes. In other words, a weighted graph is a special type of labeled graph in which the labels are numbers and here we consider only positive

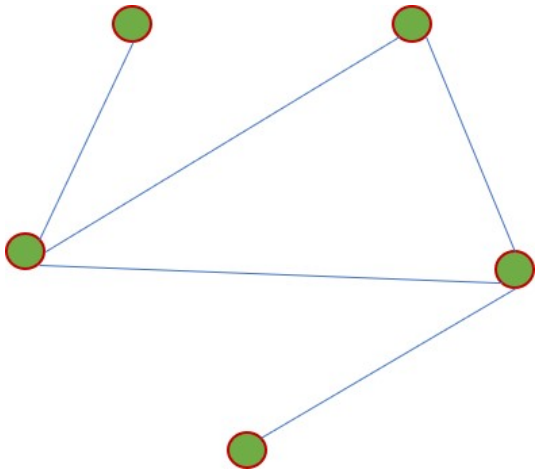


Figure 2.3: An undirected Graph

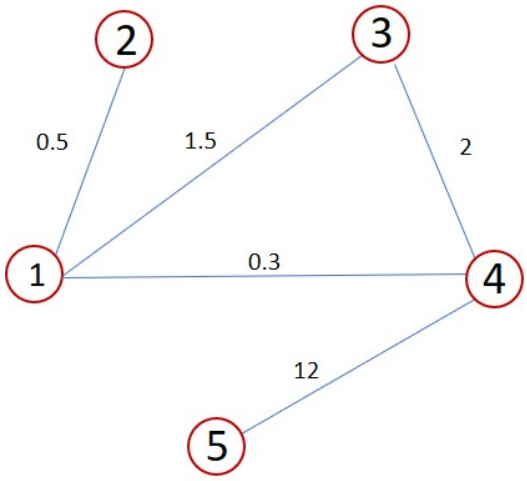


Figure 2.4: A weighted graph with numerical labels on the edges. An integer on each node illustrates its number and is used when the graph is represented by a matrix.

numbers. Figure 2.4 illustrates a weighted graph and also Figure 2.3 can be considered as a weighted graph with all weights equal to one. Such weights, for example, may represent costs of delivery between two endpoints or distance of two points depending on the problem at hand. To give a real-world example of a weighted graph, let us assume that we want to create a distance map of different cities and we show the distance of two cities by an edge with the weight in kilometers.

2.1.2.4 Complete graph

If each pair of vertices is connected by an edge, the graph is called complete (Figure 2.5). A complete graph is called K_N where N is the number of vertices. A K_N has $\frac{N(N-1)}{2}$ edges. In network topology, this type of graph is also called a ”fully con-

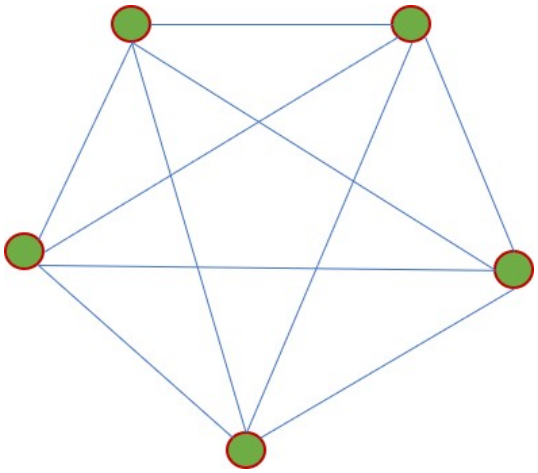


Figure 2.5: K_5: A complete graph of order 5.

nected network". One application of this graph is to initialize a topology inference algorithm when we have no a priori information about the underlying topology and thus we start with a complete graph and remove some of the edges in each iteration.

2.1.2.5 Connected graph

A graph is connected when it has exactly one connected component. In a connected graph, there is a path between every pair of vertices. Equivalently, there are no unreachable vertices in a connected graph. All examples of Figure 2.3 to Figure 2.5 are connected, but the one in Figure 2.1 is disconnected. An example of this type of graph can be the current coronavirus (Covid-19) contagion among people. If the entire contaminated population by this virus is depicted by a graph, each person must have contact with at least another one. In the bigger picture, in the beginning, each country should have at least one incoming traveler from another contaminated country. Therefore it is not possible to have a contaminated area where there is no connection with other places in the world. For another example, if we are interested in finding the shortest path between two nodes such as the shortest route between two points in the city for a cab driver, the map must be illustrated as a connected graph.

2.1.2.6 Simple graph

It is an unweighted, undirected graph containing no self-loops or multiple edges. Figure 2.3 and Figure 2.5 are examples of a simple graph. Another name for a simple graph is strict graph [59] and usually, when the term "graph" is called, it means a simple graph unless otherwise stated.

2.1.3 Matrix Representations

The node connectivities can be mathematically represented by a graph matrix, depending on the type of graph. Some of these matrices are as follows.

2.1.3.1 Adjacency matrix

This matrix shows the graph connectivities or links among nodes. The adjacency matrix $\mathbf{A}$ stores only the presence/absence of edges, regardless of their weights. In other words, this matrix indicates whether pairs of vertices are adjacent or not. If there is a connection or edge between the vertices v_i and v_j, the corresponding entity a_{ij} is one and otherwise it is zero. The adjacency matrix for an undirected graph is symmetric, but it is not necessarily symmetric for a digraph. The diagonal elements of the matrix are all zeros if the graph has no self-loop.

For some examples, the adjacency matrix of a complete graph includes all ones except the diagonal entries which are zeros. The adjacency matrix of an empty graph is a zero matrix. The adjacency matrix for the graph shown in Figure 2.4 is given as below

$$\mathbf{A} = \begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{2.2}$$

Since the adjacency matrix for an undirected graph is symmetric, it has a set of real eigenvalues and an orthogonal eigenvector basis. By applying the Perron–Frobenius theorem, the greatest eigenvalue of the adjacency matrix is bounded above by the maximum degree of the graph. There are more details about the graph spectral analysis in 2.4.

2.1.3.2 Weight matrix

This matrix stores the edge weights of a weighted graph. Each edge weight w_{ij} is stored in the corresponding entry of $\mathbf{W} \in \mathbb{R}^{N\times N}$. Thus, $w_{ij} = 0$ shows no connection and w_{ij} quantifies strength of the connection between the node v_i and the node v_j. In this dissertation, we consider non-negative weights and hence $\mathbf{W} \in \mathbb{R}_{+}^{N\times N}$ (Figure 2.4). The diagonal entries of the weigh matrix show the weight of the self-loops and thus for a graph with no self-loop, the diagonal elements of $\mathbf{W}$

are zeros. The weight matrix for the graph shown in Figure 2.4 is as follows

$$\mathbf{W} = \begin{bmatrix} 0 & 0.5 & 1.5 & 0.3 & 0 \\ 0.5 & 0 & 0 & 0 & 0 \\ 1.5 & 0 & 0 & 2 & 0 \\ 0.3 & 0 & 2 & 0 & 12 \\ 0 & 0 & 0 & 12 & 0 \end{bmatrix} \tag{2.3}$$

2.1.3.3 Degree matrix

The degree matrix is defined as $\mathbf{D} = \text{diag}(d_i), 1 \leq i \leq N$. For an unweighted graph, d_i simply counts the number of edges connected to the node i, and for a weighted graph, d_i is the sum of all edge weights connecting node i to its neighbors. Then, it can be rewritten as $\mathbf{D} = \text{diag}(\mathbf{W} \cdot \mathbf{1}_N)$. The degree matrix of the graph shown in Figure 2.4 is given as below

$$\mathbf{D} = \begin{bmatrix} 2.3 & 0 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 3.5 & 0 & 0 \\ 0 & 0 & 0 & 14.3 & 0 \\ 0 & 0 & 0 & 0 & 12 \end{bmatrix}$$

2.1.3.4 Laplacian matrix

The combinatorial graph Laplacian matrix is defined as

$$\mathbf{L} = \mathbf{V} + \mathbf{D} - \mathbf{W}, \tag{2.4}$$

where $\mathbf{V}$ stores the self-loops in its diagonal. Thus, if there is no self-loop in the graph, the graph Laplacian is reduced to

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{2.5}$$

The Laplacian matrix is also called the graph Laplacian, admittance matrix, Kirchhoff matrix, or discrete Laplacian. The Laplacian matrix for the graph in

Figure 2.4 is as follows

$$\mathbf{L} = \begin{bmatrix} 2.3 & -0.5 & -1.5 & -0.3 & 0 \\ -0.5 & 0.5 & 0 & 0 & 0 \\ -1.5 & 0 & 3.5 & -2 & 0 \\ -0.3 & 0 & -2 & 14.3 & -12 \\ 0 & 0 & 0 & -12 & 12 \end{bmatrix} \tag{2.6}$$

In this dissertation, we mainly focus on graphs without self-loops and non-negative edges. Thus, the normalized graph Laplacian is defined as

$$\begin{aligned} \mathbf{L}_{\text{norm}} =& \mathbf{D}^{\frac{-1}{2}}\mathbf{L}\mathbf{D}^{\frac{-1}{2}} \\ =& \mathbf{I}_N - \mathbf{D}^{-\frac{1}{2}}\mathbf{W}\mathbf{D}^{-\frac{1}{2}} \end{aligned} \tag{2.7}$$

and thus, the elements of the normalized Laplacian is given as follows

$$L_{ij} = \begin{cases} 1, & i = j \text{ and } \deg(v_i) \neq 0 \\ -\frac{1}{\sqrt{\left(\deg(v_i)\deg(v_j)\right)}} & i \neq j \text{ and } v_i \text{ is adjacent to } v_j \\ 0 & \text{Otherwise} \end{cases} \tag{2.8}$$

The Laplacian matrix is symmetric, positive semi-definite (PSD), diagonally dominant, and singular. Always, the first eigenvalue is zero since the eigenvector $\mathbf{v}_0 = (1, 1, \ldots, 1)$ satisfies $\mathbf{L}\mathbf{v}_0 = \mathbf{0}$. If the graph is disconnected, the number of zero eigenvalues is equal to the number of components. Thus, the number of connected components is the dimension of the null space of the Laplacian and the algebraic multiplicity of the zero eigenvalues. Thus, in a graph with multiple connected components, the Laplacian matrix is block diagonal, where each block is the respective Laplacian matrix for each component.

2.1.3.5 Incidence matrix

This matrix has a row for each vertex and a column for each edge and hence its size is $|\mathcal{V}|$ by $|\mathcal{E}|$. If a vertex is connected to an edge, the corresponding element in the incident matrix is one. A nice property of the incidence matrix is that multiplying it by its transpose results in the Laplacian matrix. The incidence matrix for graph of the Figure 2.6 is as follows

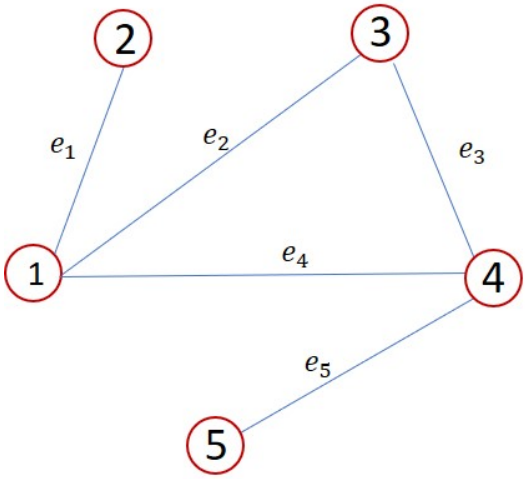


Figure 2.6: The edges are labeled to be used for incidence matrix.

$$\mathbf{B} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

If the graph is not weighted, the sum of each column is equal to two since each edge has a vertex connected to each end. If the graph is directed, the usual convention is as follows

$$B_{ij} = \begin{cases} 1 & \text{the edge } e_j \text{ enters vertex } v_i, \\ -1 & \text{the edge } e_j \text{ leaves vertex } v_i, \\ 0 & \text{Otherwise.} \end{cases} \tag{2.9}$$

2.2 Graph Topology

A graph can represent a network topology in an abstract form. The ”Topology” is the form in which the nodes and links/connections reside within a network. Usually, vertex-edge nomenclature is used to refer to the abstract mathematical graph concept and node-link to denote the topology of a real-world network. Regardless of this notation, depending on the regularity of placing vertices and edges, there are two general categories of known and unknown graph topologies.

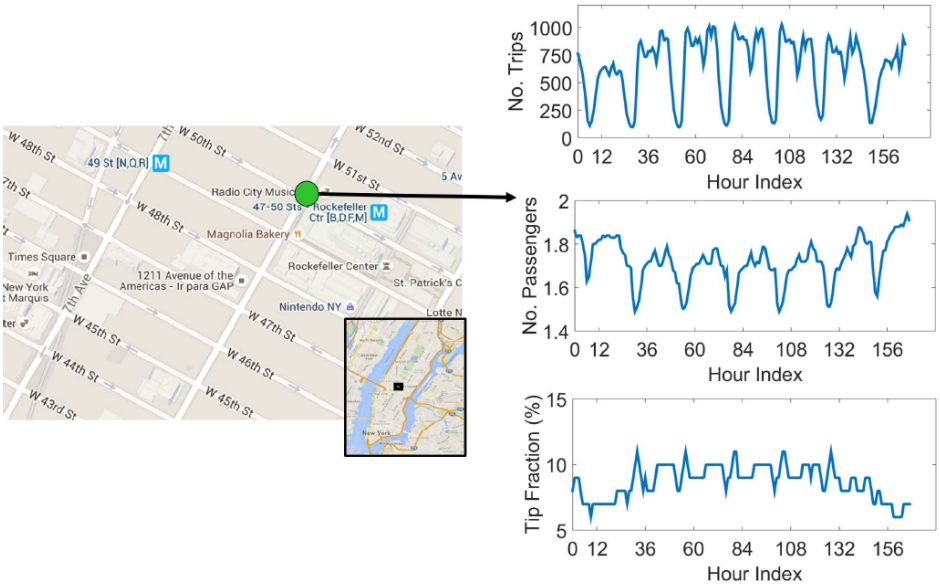


Figure 2.7: The Manhattan grid pattern [2].

2.2.1 Regular Structures

In some applications, we have a priori information about the underlying topology which is one of the well-known structures. Since the structures are known here, we can compute any matrix representation for future use. In what follows, some of these structures are introduced.

- **Grid**: This topology has a regular grid pattern. One of the examples in the real-world application is graph signal extraction for New York City taxi data [2]. Figure 2.7 illustrates Manhattan traffic activity patterns where each junction is considered as a node and the street between them is a link.

- **Tree**: It is a simple connected graph without any cycle. Thus, for a tree with N nodes, there exist $N-1$ links. One of the most useful examples is a centralized cellular network topology when the core system is the root of the network which communicates with the wireless endpoints with communication links (see Figure 2.8).

- **Star**: A tree becomes a star if only one node has a degree larger than one. For example, in Figure 2.8, consider only one hexagonal cell in which a base station communicates separately with all mobile users, but mobile users can not directly talk to each other.

- **Chain**: A chain is a tree with no nodes of a degree greater than two. For example, assume the Manhattan grid pattern of Figure 2.7 where a taxi has a

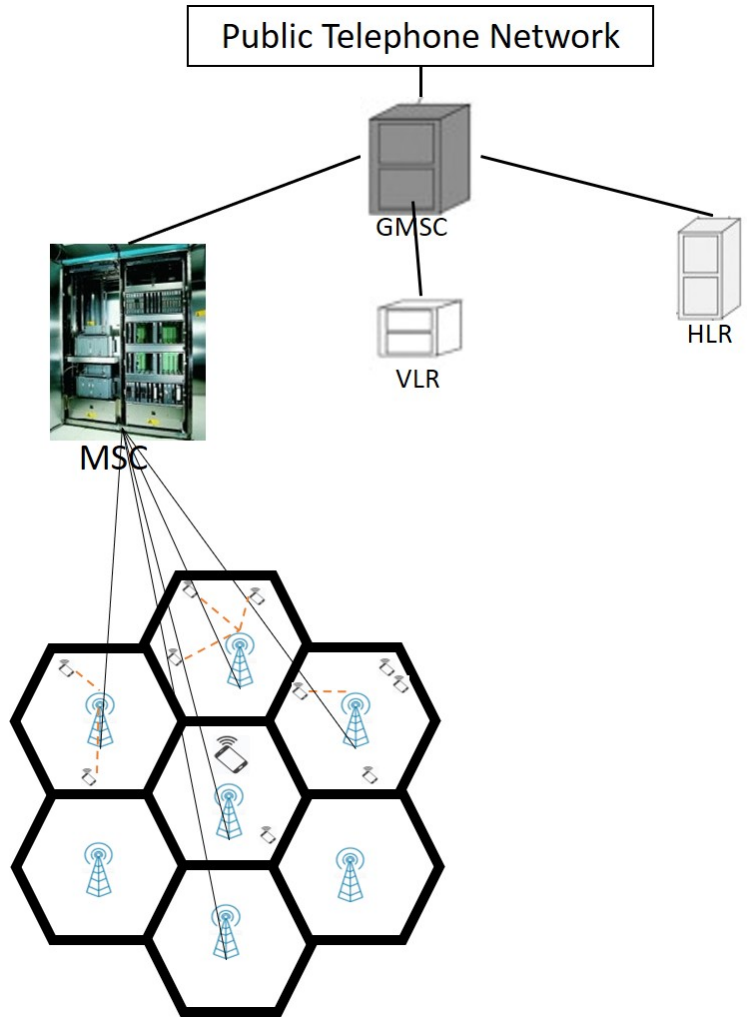


Figure 2.8: A wireless mobile network with a tree structure.

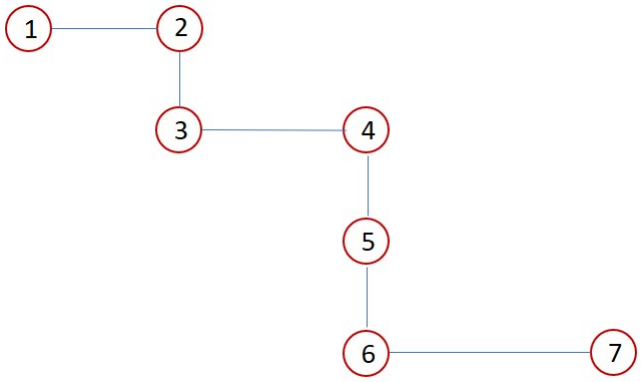


Figure 2.9: The route of a taxi as the chain topology.

pick-up location at one point and drop-off at another point. The route makes a chain as Figure 2.9.

- **Ring**: Given a set of N nodes, a tour or a ring is a set of N links such that the graph is connected and each node has degree two. This topology is famous due to the well-known "traveling salesman problem" which is as follows: given a set of cities and all the distances, it is desired to find the shortest route visiting each city exactly once and returns to the origin.

2.2.2 Irregular Structures

The simplest and widely used methods to find some types of structure or topology are based on a similarity or correlation, and some functions of them. These methods are mainly focused on pairwise relationships of data. In contrast, the GSP framework proposes to investigate a direct relationship between data sets and the data structure in general which we will discuss at the end of this subsection shortly (and then the next chapters of this thesis mainly focus on some graph learning procedures based on it). In general, a graph topology can take any structure that is not among the well-known ones. Depending on the criteria of the problem at hand and a priori information, the underlying structure can be estimated via some simple algorithms. In this subsection, some of these estimation methods are introduced. Here, the appropriate matrix representation is computed first and then the graph topology is built. Before introducing those methods, let us review some examples.

When the relationship between two nodes is of interest, but it is not important whether the connection is weak or strong, it is possible to find and represent the topology by a graph adjacency matrix, such as Twitter followers/following relationships example in Sec. 2.1.3. On the other hand, if the problem involves the

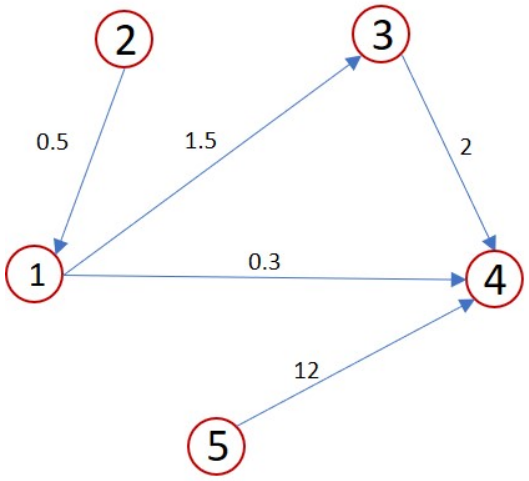


Figure 2.10: A directed graph that has a similar topology to the one in Figure 2.4.

importance of the strengths of each connection, we have to be more specific and solve a more complex and harder problem in which the weight matrix is estimated. For an undirected graph, the Laplacian matrix can also uniquely characterize the graph structure. To clarify more these statements and conditions, an example of weighted and directed graph is given in Figure 2.10 and its corresponding matrices is computed in (2.10) which can be compared with 2.2, 2.3, and 2.6.

$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad \mathbf{W} = \begin{bmatrix} 0 & 0 & 1.5 & 0.3 & 0 \\ 0.5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12 & 0 \end{bmatrix}$$
$$\mathbf{L} = \begin{bmatrix} 1.8 & 0 & -1.5 & -0.3 & 0 \\ -0.5 & 0.5 & 0 & 0 & 0 \\ 0 & 0 & 2 & -2 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -12 & 12 \end{bmatrix} \tag{2.10}$$

2.2.2.1 Neighborhood graph

After setting a pre-defined threshold, the distance between each pair of the nodes is computed. If this distance is less than a threshold, we connect two nodes by an edge. To compute the distance, a function based on the application is selected. The radial basis function (RBF) and Gaussian RBF are widely used functions to find the

distance. This procedure leads to an undirected graph topology since the distance of the node i to the node j is the same as the distance of the node j to the node i.

2.2.2.2 K-Nearest Neighbor

If the vertex i is among the K nearest neighbors of vertex j, there is an incident edge from node i to node j. In this way, we have a directed topology, since this definition of neighborhood relationship is not symmetric. There is a simple undirected version for the K-NN similarity graph in which we form the graph and then remove the direction. In other words, either v_i is in the neighborhood of v_j or vice versa, these two vertices are connected by an edge. There is also another approach to make an undirected K-NN called "mutual K-NN" [60]. In this case, v_i is among the K nearest neighbors of v_j and also v_j is among the K nearest neighbors of v_i. After finding the adjacency matrix by any of these definitions, the edges are weighted by a distance function or a similarity function, e.g. the inner product of the vertices coordinates.

2.2.2.3 Correlation

The correlation is used to define the distance between two vectors of data and captures the similarity of two data variables. It shows how the data vectors are close to each other and to what extent they can be related to each other. To define the correlation mathematically, first, the sample covariance is defined as follows

$$\mathbf{S} = \frac{1}{K-1} \sum_{k=1}^{K} \big(\mathbf{x}[k] - \overline{\mathbf{x}}\big)\big(\mathbf{x}[k] - \overline{\mathbf{x}}\big)^T, \tag{2.11}$$

and the ij'th element of the correlation matrix is the normalization of sample covariance entities as below

$$c_{ij} = \frac{s_{ij}}{\sqrt{s_{ii}s_{jj}}}. \tag{2.12}$$

The correlation coefficient takes on a value on the interval $[-1, 1]$, where –1 shows a perfect negative correlation, i.e. an exact linear relationship when a higher value of one variable corresponds to a lower value of the other. Similarly, the value one conveys an exact positive linear relationship. When the correlation coefficient is zero, there is no linear relationship.

If $c_{ij} \neq 0$, an edge is considered between v_i and v_j. Sometimes when it is above a threshold, the corresponding two nodes are connected by an edge with the weight c_{ij} [61]. Figure 2.11 illustrates a perspective for the correlation concept

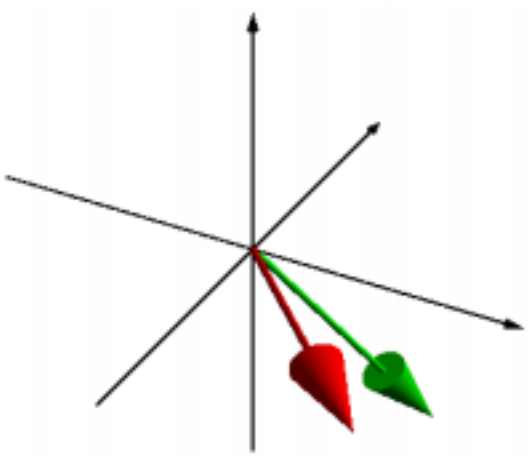


Figure 2.11: A geometrical intuition behind correlation in three-dimensional space.

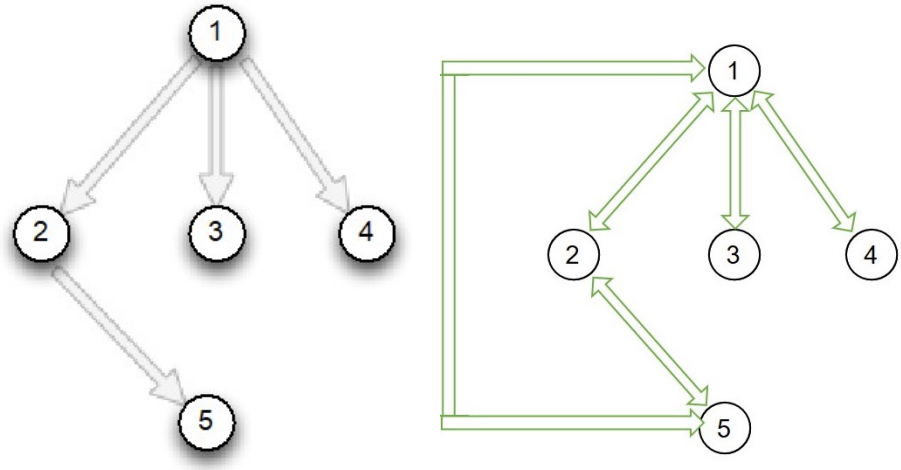
where we can consider two data points as the multidimensional points in the space. The similarity of these two points is equivalent to how much they are correlated. If they have a correlation of one, they should be completely similar and hence point toward the same location. If they have a bit of deviation, there is a small angle between them and hence the correlation is less than one.

However, the correlation method has the following limitations

- Correlation can only track the undirected relationship. For example, in Figure 2.12a, there are directed connections from node 1 to nods 2, 3, and 4. However, the correlation captures all these connections as bi-directed (see Figure 2.12b),
- It can not distinguish mediated vs. un-mediated dependencies. For example in Figure 2.12, there is no connection between nodes 1 and 5 in the true topology, while the learned topology shows that there is an edge between them. This is due to the fact that when node 1 is correlated with node 2, and node 2 is also correlated with node 5, a correlation between nodes 1 and 5 is not avoidable,
- The correlation method is very sensitive to the noise. In noisy conditions, the performance of the edge recovery is low [41].
- The correlation concerns only about the linear relationships. In some applications, we want to capture any kind of relationship and connectivities.

2.2.2.4 Partial Correlation

To solve the problem of mediated dependency in correlation method, the partial correlation can be applied. Actually, it measures the degree of relationship between



(a) The real topology (b) The estimated topology by correlation

Figure 2.12: Wrong network topology learning by correlation approach.

two random variables when the effect of a third random variable is removed. As is shown in Figure 2.12, the use of the correlation concept gives misleading results if there is another variable, numerically related to both variables. For example, if we have a data-set of consumption, income, and wealth of people and want to capture how the consumption is correlated with the wealth, the partial correlation works better than the correlation. Actually, the partial correlation measures the strength of the linear relationship between consumption and income after "adjusting" for the relationship involving wealth. As an illustration, if the interest is in the blue rectangular in Figure 2.13, the partial correlation is a good candidate.

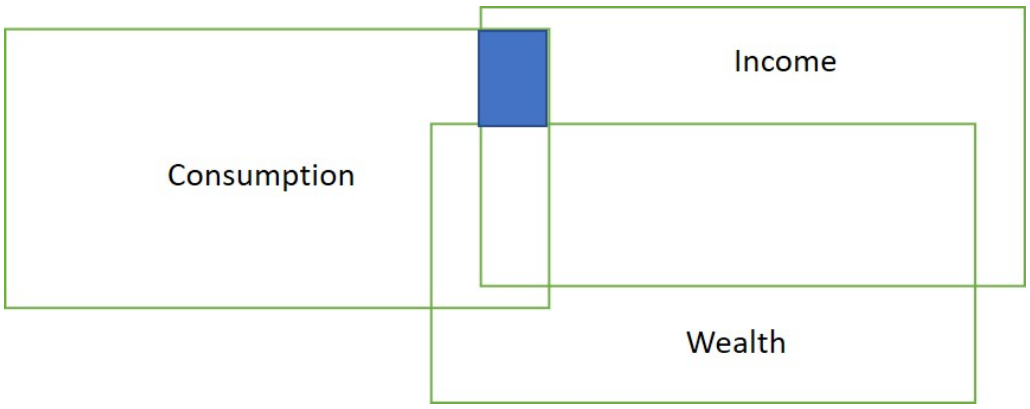


Figure 2.13: The correlations among consumption, wealth, and income. The partial correlation captures the blue rectangular.

The partial correlation is found simply from the inverse covariance or precision

matrix as follows [62]

$$\rho_{ij} = -\frac{s_{ij}^{-1}}{\sqrt{s_{ii}^{-1} s_{jj}^{-1}}}. \tag{2.13}$$

The partial correlation is able to solve the issue of mediated or un-mediated dependency, but it has all the other issues of the correlation method.

2.2.2.5 Learning Dynamic Connectivities

The main intention of this approach is to focus on the global behavior of signals residing on nodes instead of focusing on pairwise relationships of two nodes. To do so, some methods have been investigated in the GSP framework to relate the multivariate signal to the underlying structure, e.g. [19, 41, 49, 50, 63–66] and so on. In other words, here, we are interested in representing the graph signals as a function of the underlying topology. This is the main topic of this thesis and thus in the next chapters, we discuss this approach for different scenarios and conditions.

2.3 Random Graph Models

There are several ways to generate a graph at random where we need to model different types of networks. Also, if a random graph is required to generate some graph signals for algorithm evaluations, this is a useful tool. Here, we represent the most important types of random graph generation.

2.3.1 Erdős-Rényi

Most of the time, this type of graph is the default random graph generator and hence it is sometimes called "the random graph". The Erdős-Rényi graph is denoted by $\mathcal{G}(N, p_0)$ where N is the number of nodes and $0 \leq p_0 \leq 1$ denotes the probability of connecting any two nodes in the graph. To generate an Erdős-Rényi, N nodes is considered, i.e. $v_i, \quad i = 1, \quad \ldots, \quad N$, and then for each unique pair of v_i and v_j, a biased coin is flipped where the probability of head is p_0. If it hits the head, an edge is created between v_i and v_j, and for the tail event, we let these two nodes be disconnected. Thus, every time we generate this graph, we get a different one due to the random nature of tossing coins, and hence $\mathcal{G}(N, p_0)$ is a family of graphs. For example, assume that $N = 3$ and hence there are 8 possibilities of tossing a coin and hence 8 graphs as depicted in Figure 2.14.

The maximum number of possible edges is $\binom{N}{2}$ and hence it is most probable to have those graphs with the number of edges about $p_0 \cdot \binom{N}{2}$, i.e. average number of

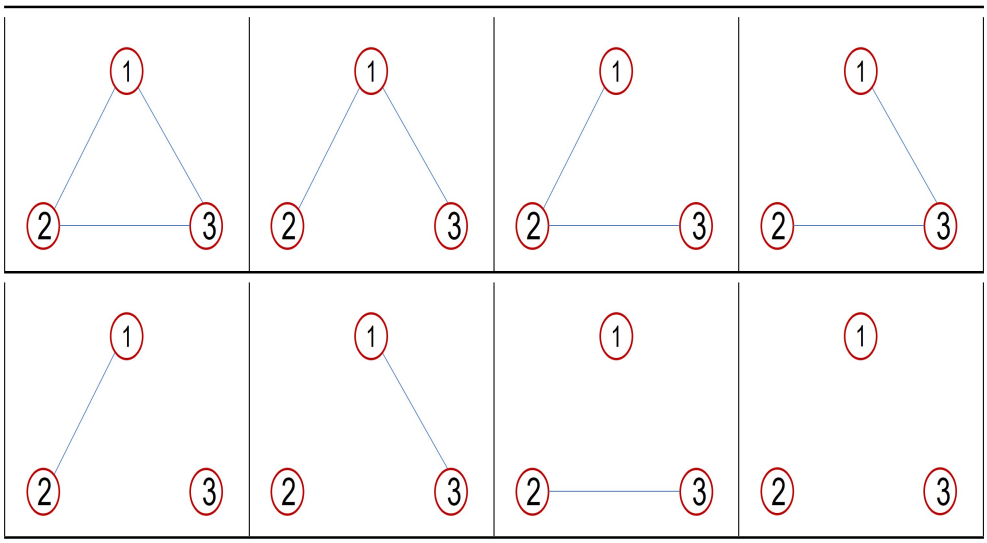


Figure 2.14: The Erdős-Rényi family graphs with $N = 3$ vertices.

edges. The probability of generating a specific graph in Figure 2.14 is as follows

$$P(\mathcal{G}) = p_0^m (1 - p_0)^{\binom{N}{2} - m}, \tag{2.14}$$

where m is the number of edges in the graph. In other words, this is the probability of having a specific sequence of heads and tails in coin flips. For example, the probability of having a graph with 3 edges when $N = 3$, equals p_0^3. This is a simple Bernoulli probability density function. In the same way, the probability of generating a graph with m edges follows a Binomial distribution given as below

$$P(m) = \binom{N}{2} p_0^m (1 - p_0)^{\binom{N}{2} - m}. \tag{2.15}$$

2.3.2 Barabási–Albert

An aphorism states that ”rich get richer” which can be assumed the cornerstone of the preferential attachment process in network science. A preferential attachment is a class of processes in which some quantities are divided among different objects based on how much they already have. As an example, if some kinds of wealth are distributed among people, those who are already wealthier receive more than those who are poorer. In network sciences, the main motivation for interest in preferential attachment is its ability to generate power law distributions. The first application of preferential attachment belongs to Udny Yule in 1925 [67] where the power-law distribution of the number of species per genus of flowering plants has

been explained. Thus, this process is sometimes called a "Yule process" instead of preferential attachment.

The Barabási–Albert graph model is based on a preferential attachment process and generates an scale-free network. There are many applications for this type of network, e.g. the world wide web, social networks, and citation networks. In a social network, a famous actor or politician can get more and more followers compared to a typical individual. In the same way, in the citation network, a scientist with higher citations and a larger number of papers gets easily more citations over a specific time interval. Therefore, this graph model contains few nodes with unusually high degrees. In simple words, the preferential attachment means that a more connected node is more likely to receive new links. In addition to this property, the BA model has another main property which is called growth, meaning that the number of nodes in the network increases over time.

To sample a BA graph model, an initial network is generated. A new node is linked to the existing node i with the probability of $p_i = \frac{n_i}{\sum_j n_j}$, where n_i is the degree of node v_i and the summation is over all existing nodes v_j. This probability is proportional to the number of connections that the existing nodes already have and a new node has a "preference" to attach itself to the already highly connected nodes.

2.4 Graph Spectral Analysis

For undirected graphs with non-negative edges and no self-loops, the Laplacian matrix is real and symmetric and then its eigendecomposition is as follows

$$\mathbf{L} = \boldsymbol{\chi}\boldsymbol{\Lambda}\boldsymbol{\chi}^T, \tag{2.16}$$

where in the diagonal entries of $\boldsymbol{\Lambda}$, eigenvalues is always ordered increasingly, respecting their multiplicities. By referring to the first n_1 eigenvalues, the n_1 smallest eigenvalues are aimed. The graph Laplacian matrix has a complete set of eigenvectors, which are the columns of $\boldsymbol{\chi}$ defined in (2.16). Moreover, the eigenvalues are real and non-negative, denoted here by $0 = \lambda_0 \leq \lambda_2 \leq \cdots \leq \lambda_{N-1}$. The first eigenvalue, i.e. λ_0 is zero since each row (or each column) of $\mathbf{L}$ sums to zero. For a connected graph, there is at least one zero eigenvalue. In general, a graph has at least n_0 zeros eigenvalues if it has n_0 separate components. The eigenvector corresponding to the zero eigenvalue is constant and equal to $1/\sqrt{N}$ at each vertex. Thus λ_0 is analogous to the zero frequency in the classical signal processing.

Similar to the Fourier transform (FT) which helps to represent complex signals

by their fundamental frequencies, the spectral graph synthesis and analysis decompose graph signals with respect to the underlying graph and spectral characteristics of the signals. The Graph spectral analysis is done not only based on the signal itself but also considering the connection among the signal vector entities. Thus, the graph Fourier transform (GFT) is defined with the help of eigendecomposition of the graph Laplacian. The main difference between FT with GFT is the importance of the underlying topology for the GFT. In other words, not only the graph signal characteristics are important for GFT computation, but the graph topology is also involved.

A graph signal is a multivariate signal, residing on vertices of the graph. When the graph has N vertices, the graph signal is an N-dimensional vector. A graph signal $\mathbf{x}$ at time instant k is given as below

$$\begin{aligned} &\mathbf{x}[k] : \mathcal{V} \to \mathbb{R}^N, v_i \mapsto x_i[k] \\ &\mathbf{x}[k] = \big(x_1[k], x_2[k], ..., x_N[k]\big)^T \in \mathbb{R}^N, \end{aligned} \tag{2.17}$$

where $x_i[k]$ is the signal value at node i and time k. The GFT of the graph signal $\mathbf{x}$ is defined as follows [3]

$$\text{GFT}_{\mathbf{x}}(\lambda_l) := \langle \mathbf{x}, \boldsymbol{\chi}_{:l} \rangle = \sum_{i=1}^{N} x_i \chi_{il}. \tag{2.18}$$

Thus, the inverse GFT is given by

$$x_i = \sum_{l=0}^{N-1} \text{GFT}_{\mathbf{x}}(\lambda_l) \chi_{il}. \tag{2.19}$$

The eigenvectors associated with low frequencies change slowly within the graph. In other words, if there is a very strong edge connecting two vertices, the values of the eigenvector at those vertices are close [3]. This property is also called "signal smoothness" over the underlying graph. In opposite, an eigenvector associated with a large eigenvalue changes sharply and it has dissimilar values on strongly connected nodes. This phenomenon are illustrated in Figure 2.15 for a sample sensor network graph in [3]. As is shown, the first eigenvector has no zero-crossing and all the values are above zero and similar. Thus, it is a kind of DC component and completely smooth over the graph. The second eigenvector of the graph called Fiedler eigenvector changes smoothly from zeros to ones, in a limited area in the middle of the graph. The last graph in Figure 2.15 shows the 50'th eigenvector corresponding to a large eigenvalue and changes sharply from one node to another and

oscillating from one to zero and vice versa. For a graph signal $\mathbf{x}[k]$, we have

$$(\mathbf{L}\mathbf{x}[k])_i = \sum_{j \in \mathcal{N}_i} w_{ij} \big[\mathbf{x}_i[k] - \mathbf{x}_j[k]\big], \tag{2.20}$$

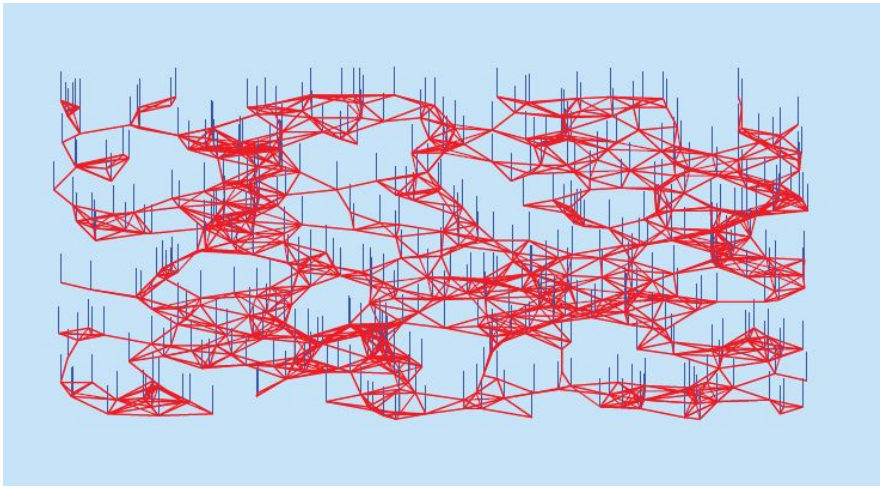
where $\mathcal{N}_i$ is the neighborhood of node i, i.e. the set of vertices connecting to node i by the existing edges. The local smoothness around vertex i is computed as [3]

$$\|\nabla_i \mathbf{x}[k]\|_2 := \Big[\sum_{j \in \mathcal{N}_i} w_{ij} \big[\mathbf{x}_i[k] - \mathbf{x}_j[k]\big]^2 \Big]^{\frac{1}{2}}, \tag{2.21}$$

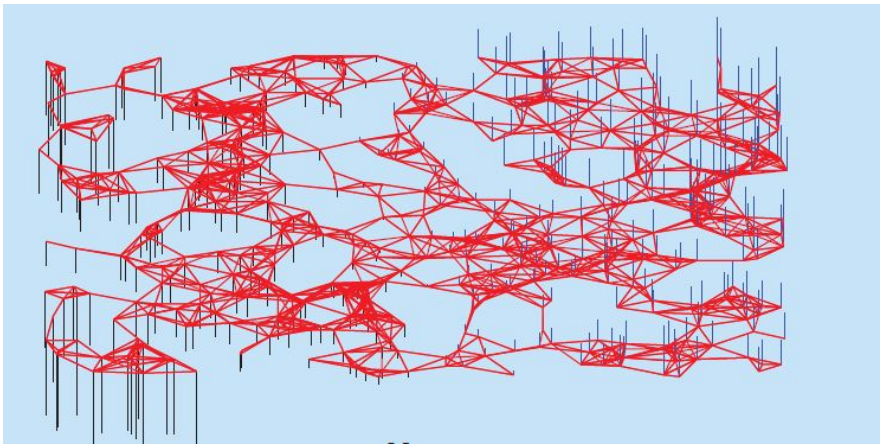
and the global smoothness is defined as follows [68]

$$S_2(\mathbf{x}[k]) = \sum_{j \in \mathcal{N}_i} w_{ij} \big[\mathbf{x}_i[k] - \mathbf{x}_j[k]\big]^2 = \sum_{(i,j) \in \mathcal{E}} \sum_{j \in \mathcal{N}_i} w_{ij} \big[\mathbf{x}_i[k] - \mathbf{x}_j[k]\big]^2 = \mathbf{x}[k]^T \mathbf{L}\mathbf{x}[k]. \tag{2.22}$$

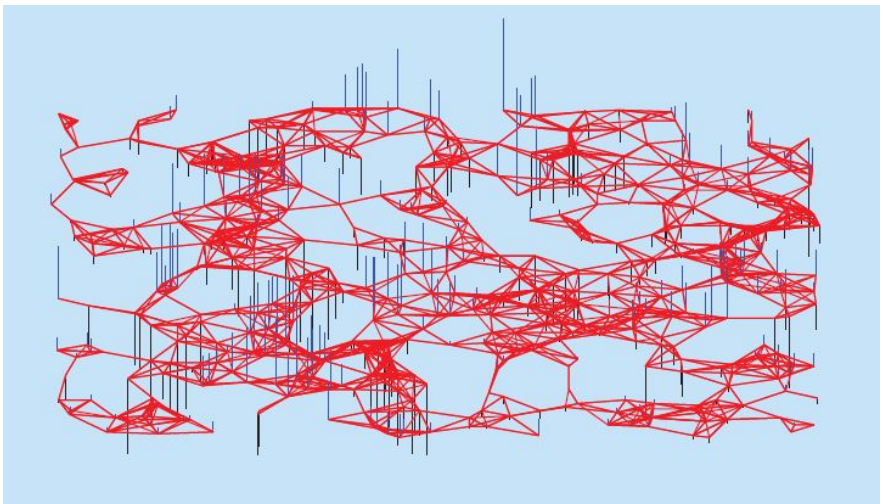
In Figure 2.16, the signal $\mathbf{x} = [-3, -1, 1, 3, 5]$ is shown on three different graph structures to compare smoothness visually. Assuming all existed edges have a weight equal to one, the measure of 2.22 for the signal over these three graphs are computed as 16, 80, 120, respectively. From the numerical point of view, the graph signal is smoother with respect to $\mathcal{G}_1$ when compared to $\mathcal{G}_2$ and also smoother with respect to $\mathcal{G}_2$ when compared to $\mathcal{G}_3$.



(a) The DC component, i.e. $\boldsymbol{\chi_{:0}}$, corresponding to λ_0

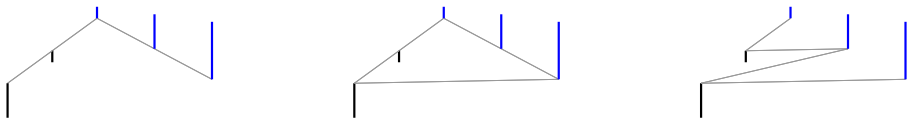


(b) The Fiedler eigenvector, i.e. $\boldsymbol{\chi_{:1}}$, corresponding to λ_1



(c) The 50'th eigenvector, i.e. $\boldsymbol{\chi_{:50}}$, corresponding to λ_{50}

Figure 2.15: Different eigenvectors of a sample graph which illustrate visually how they are changing on nodes [3]. The blue and black bars show positive and negative values, respectively. The graph edges are shown by red connections.



(a) $\mathbf{x}$ over $\mathcal{G}_1$ (b) $\mathbf{x}$ over $\mathcal{G}_2$ (c) $\mathbf{x}$ over $\mathcal{G}_3$

Figure 2.16: A unique graph signal resided on three different graph topologies (The figures are taken from [4]). The signal is smooth with respect to $\mathcal{G}_1$, less smooth with respect to $\mathcal{G}_2$ and likely to be non-smooth with respect to $\mathcal{G}_3$. The edge weights are all ones. These figures are generated by GSPBOX [5].

2.5 Graph Filters

A graph filter is a system taking a graph signal as an input and making another graph signal in the output indexed by the same graph vertices. In the classical signal processing, the basic block for filters is a shift operator $\mathcal{S}$, represented as [16, 69]

$$\mathbf{x}_{\text{shifted}} = \mathcal{S}\mathbf{x}, \tag{2.23}$$

and the general form of the linear shift invariant (LSI) filter is

$$\mathbf{x}_{\text{filtered}} = \mathcal{S}_1\mathbf{x}[k] + \mathcal{S}_2\mathbf{x}[k] + \cdots + \mathcal{S}_{N_S}\mathbf{x}[k]. \tag{2.24}$$

In the simplest case, $\mathcal{S}$ is the adjacency matrix, the weight matrix, or the graph Laplacian. However, in general, it can be any function of the graph topology matrices. If we let the shift operator be the weight adjacency matrix and H denotes the filter, the graph LSI filter is defined as follows

$$\mathbf{x}_{\text{filtered}} = H(\mathbf{W})\mathbf{x}[k] = (h_0\mathbf{I} + h_1\mathbf{W} + ... + h_{N_W-1}\mathbf{W}^{N_W-1})\mathbf{x}[k] \tag{2.25}$$

where h_0, h_1, ..., h_{N_W} are the filter coefficients and N_W is the number of filter taps (filter order). If there are two graph filters H_1 and H_2, the following is held by linearity

$$(a_1H_1 + a_2H_2)\mathbf{x} = a_1H_1\mathbf{x} + a_2H_2\mathbf{x}, \tag{2.26}$$

for some constants a_1 and a_2. By the shift invariance property,

$$\mathcal{S}\big(H\mathbf{x}\big) = H\big(\mathcal{S}\mathbf{x}\big) \tag{2.27}$$

2.6 Chapter Summary

This chapter reviews some definitions and concepts of graph theory and connects them to the area of graph signal processing. The GSP framework mainly focuses on multivariate signals residing on a graph topology. We also discussed the network topology and graph signals so as the simple existing methods for learning the graph topology from the data. Also, the concept of Fourier analysis is introduced to the graph signals, enabling the spectral analysis of multivariate signals based on the underlying data structure.

Chapter 3

Undirected Topology Learning via Bayesian Inference

The focus of this chapter is on one of the most important graph process models, called Gaussian Markov Random Field (GMRF) processes where the estimated topology is undirected. A graph topology learning algorithm is proposed to estimate the underlying structure of the given set of noisy multivariate observations generated from a GMRF process. A factor analysis model is applied to represent the graph signals in a latent space where the basis is related to the graph topology. Thus, the given data is connected and related to the underlying topology. A minimum mean square error estimator is used for designing an optimum graph filter to recover the signal from noisy observations. In the final step, an optimization problem is proposed to learn the underlying graph topology from the recovered signals. A fast algorithm to solve the optimization problem is also proposed via the proximal point method.

3.1 Introduction

The Gaussian process model is a general signal model applied to a broad range of applications. By applying a data set generated from Gaussian processes, the solution of a topology inference problem usually leads to a graph with an undirected structure. One of the pioneering works to find the connectivity among the Gaussian measurements is "covariance selection" by Dempster [38]. Banerjee et. al. [39] proposed an ℓ_1 regularized optimization problem to find the sparse precision matrix. The precision matrix is the inverse of the covariance matrix and carries information

about the partial correlations of random variables. Friedman and his colleagues have proposed "graphical Lasso" to estimate the inverse of covariance [42]. Also, some research problems and solutions are investigating the inverse covariance matrix estimation with different rates of convergence [45,46]. However, all of these works can not fully connect the signal model to the underlying graph topology. Their proposed approaches have focused on pairwise relationships instead of the global signals and graphs connections and were very sensitive to noise.

A state-of-the-art machine learning approach to estimate the graph topology from the given Gaussian Markov Random Field (GMRF) processes measurements was proposed in [40]. The main limitations were high-cost implementation for a large number of nodes and the availability of noise-free measurements. In another approach, an efficient and scalable algorithm graph learning from noise-free observations has been proposed for positive node degrees topologies (they can not be zero) [43]. A generalization of these methods for different Laplacian and constraints has been proposed in [53]. In [44], by applying graph signal frequency domain analysis, the network topology has been inferred from spectral templates. It was assumed that a diffusion process in the graph shift operator (GSO) generating the given observations exists. By capturing the graph's local topology, the GSO matrix eigenvectors devise the graph Fourier transform bases. When the graph signals are generated from a white process given as the input of the GSO filter, this method is utilized to find the topology. Other similar approaches have been presented in [47–52], There are many more papers that investigate the topology estimation and graph learning methods for different perspectives [32, 64, 70–75].

However, none of these approaches has discussed graph signal recovery and topology inference for noisy observations. Removing the noise from entries of a corrupted data matrix is desired in many applications [76] while recovering a more accurate learned topology. Sometimes the measurement errors in multivariate data are modeled as noise [77]. One assumption is that the measurement noise is a zero mean Gaussian and independent of the signal of interest. Therefore, finding the noise variance is of interest to design a filter for signal denoising and have better quality observations. If we investigate this problem from a mathematical perspective, to overcome the over-fitting issue in the optimization problem, the data fidelity term is regularized by a smoothing term and an exhaustive search is applied to find the regularization parameter. In many applications of multivariate data analysis, a direct relationship between the noise variance and the regularization parameters is held, e.g. image restoration [78]. In other words, finding the measurement noise's variance can also tackle the over-fitting problem in a noisy environment so as a bet-

ter signal representation. The noise variance estimation is the cornerstone of a filter design to smooth the signal and denoise the measurements [1].

Not only the noise variance estimation is useful from the signal recovery perspective, but also it is important from the graph learning side. Inferring the structure from a noise-free data set is more reliable than that of noisy measurements. Therefore, signal denoising helps to reach a more accurate graph topology estimation. Sundeep et al. [79] investigated the graph connectivity and removed noise from observations by solving a non-convex cardinality constrained optimization problem. It is assumed that the number of edges is known as a priori and their proposed method scales with the desired number of edges due to the use of a sorting algorithm. Another approach is proposed in [41] estimating the topology and removing the noise from measured signals in parallel. A factor analysis model has been adopted for the multivariate signals and a Gaussian prior was imposed on the latent variables that control the graph signals. By applying the Bayesian framework, the posterior distribution of graph signals was achieved as Gaussian and then the latent variable is estimated by the maximum a posteriori (MAP) approach. Considering signal smoothness on the graph, this procedure ended in an optimization problem for the graph topology and graph signals. However, in [41], an exhaustive search is applied to find the regularization parameter or noise variance which can be improved by an analytical and a more exact mathematical solution.

Our main contribution in this chapter is as follows; Given a set of noisy multidimensional signal observations, a new algorithm is proposed to jointly find the underlying topology and recover the graph signals. This graph structure models the affinity relationship among the multivariate data vectors. From the perspective of simultaneous graph learning and signal recovery via a Bayesian inference approach, the work most relevant to ours is the one by Dong et al. [41]. However, we proposed a minimum mean square estimation (MMSE), leading to the topology learning, graph signal recovery, and analytical noise variance estimation, simultaneously. We are not running a grid search for estimating the regularization parameter. As we discussed above, this regularization parameter also plays the role of the filter coefficient to denoise the graph measurements. To clarify more, the MMSE procedure is utilized to estimate the optimum Laplacian matrix, whose eigenvalue matrix is the precision matrix of the GMRF process. At the same time, the proposed algorithm filters the given measurements to find the desired signal. The simulation

[1] Hence in our problem setup, the concepts of filter coefficient, the noise variance, and the regularization parameter of the smoothing term in the optimization problem are tightly connected and estimating one of them leads to the others. Therefore, we may interchangeably use these three concepts hereafter.

results corroborate the higher performance of our proposed algorithm compared to the state-of-the-art algorithms.

The rest of this chapter is organized as follows; Section 3.2 formulates the graph topology learning problem via a Bayesian framework and proposes a general algorithm for implementation. In section 3.3, a method is proposed to implement Bayesian topology learning (BTL) efficiently via solving a primal proximal point algorithm, called BTL-PPA whose convergence is also proven. Sections 3.4 and 3.5 discuss the convergence of the algorithm and the experimental results, respectively. This chapter is an extension of the work in [4] and [83].

3.2 Bayesian Topology Learning

In the first subsection, we introduce some concepts which are necessary to follow the contents of this section. The details of our proposed algorithm for topology inference via the Bayesian framework come in next.

3.2.1 Backgrounds

We introduce the basics of the Bayesian inference in the signal processing framework which is the main mathematical tool for this chapter. We also explain the concept of latent variable and factor analysis model. At the end of this subsection, we review the GMRF process which is the generative process underlying the signal model. In the next sections, these concepts are reviewed by mathematical details and utilized to implement our idea.

3.2.1.1 Bayesian Inference

In statistical inference, ”Bayesian” is on the opposite side of ”Frequentist”. In the frequentist view, the underlying truth and available data are only used to do the inference. However, in the Bayesian view, we incorporate our subjective beliefs about the parameters of interest. These prior assumptions can be highly informative such as ”the parameter is sparse” or with less amount of information, e.g. ”the parameter pdf is uniformly distributed over all possible values”. In other words, in the Frequentist method, the parameter is considered deterministic and hence we are after finding a fixed value. However, in the Bayesian method, the parameter of interest is considered probabilistic and thus it is treated as a random variable with its own prior and posterior pdfs, and thus it may have its own parameters, e.g. mean and variance, called hyperparameter.

The Bayesian inference framework applies Bayes's theorem to update the prior belief about a variable. Bayes's theorem uses recent evidence or new information to estimate the posterior distribution. The main application of the framework is for the time that a sequence of observations exists and we are interested in estimating the pdf of the underlying variable. The Bayes's rule is presented as follows

$$P(a \mid b) = \frac{P(b \mid a) \cdot P(a)}{P(b)} \tag{3.1}$$

In simple words, Bayes's rule says the "posterior" is the "likelihood" times "prior". In mathematical language, the prior belief about a, i.e. $P(a)$, as well as the likelihood $P(b \mid a)$ are required. To do the Bayesian inference, we apply this rule to update the parameters of interest based on the observations as follows

$$p(\upsilon \mid \mathbf{O}, \varrho) = \frac{p(\mathbf{O} \mid \upsilon, \varrho)p(\upsilon, \varrho)}{p(\mathbf{O} \mid \varrho)p(\varrho)} \tag{3.2}$$

where $\mathbf{o}$ is a vector of values or observations and $\mathbf{O}$ is a set of observed data points. Also, υ is the parameter (vector of parameters) of the data distribution, and ϱ is the hyperparameter or the parameter of the parameter's distribution, i.e. $\upsilon \sim p(\upsilon \mid \varrho)$.

So far, the posterior probability distribution based on the Bayesian approach is reviewed, but here is the question: why is it important to estimate it? Using a posterior distribution to predict the unknown variables has a better performance when compared with the prior distribution since some kinds of information have been used to reach the posterior. The posterior median or mean is one of the well-known robust estimators [80] for the parameter estimation in this framework defined as follows

$$\hat{\upsilon} = \mathbb{E}[\upsilon] = \int \upsilon \cdot p(\upsilon \mid \mathbf{O}, \varrho) d\upsilon. \tag{3.3}$$

3.2.1.2 Latent Space Representation

Latent space denotes an abstract space containing features that we cannot explain directly, although encoding a meaningful internal representation of the observed data. For a simple though not an exact example, consider the English language and a computer language in the "machine space". A human can understand a letter like "A" on the keyboard while a processor unit understands its binary or Ascii code version in a computer language. Here, an external event of human language is mapped to another space that is understandable for the computer. For a more technical example, we can think of the "word embedding space" in the machine learning toolboxes when running a natural language processing application. It consists of

word vectors where similar words map to vectors that lie close to each other in the embedding space. In mathematics, large differences in the observed space may be due to some small variations in a latent space or there is a tool in a latent space that helps to model the signal with a better sense. These motivate representing the variables in the latent or hidden space. In statistics, latent or hidden variables are not directly observed but inferred from the observed variables, and this procedure is called latent variable models.

One of the approaches to provide the latent space is the Factor Analysis (FA) model when a statistical method is used to explain variability among observed correlated variables using a lower number of unobserved/hidden variables. Here, these latent variables are called factors, and the observed variables are modeled as linear combinations of them. FA is mathematically viewed as a generalized technique for dimensionality reduction and it is widely used in behavioral sciences, recommender systems, and social sciences. It is especially applied when a set of observed variables shows a systematic interdependence and by helping the factors, a commonality among data is extracted.

3.2.1.3 Gaussian Markov Random Field

In this chapter, we assume that the observed data has been sampled from a multivariate Gaussian distribution, generally represented by $\mathbf{x} = \mathcal{N}\big(\boldsymbol{\mu}, \boldsymbol{\Sigma}\big)$. The relation of this distribution to the graph topology is that we assume its covariance matrix is the inverse of the graph Laplacian matrix. Hence, a GMRF's distribution, or equivalently the relation between the observed graph signals and the underlying topology, is defined as follows

$$p(\mathbf{x} \mid \boldsymbol{\Sigma}^{-1}) = \frac{1}{(2\pi)^{\frac{N}{2}} \mid \boldsymbol{\Sigma} \mid^{\frac{1}{2}}} \exp\big(-\frac{1}{2} \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x} \big). \tag{3.4}$$

The term $\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \mathbf{x}$ in (3.4) is the smoothness measure defined in (2.22). If a larger weight is given to the edge between v_i and v_j, the squared difference between corresponding data points, i.e. x_i and x_j is smaller and hence the exponential term in (3.4) is going to be reduced and thus the probability of having that data vector $\mathbf{x}$ given the topology increases. This probabilistic interpretation brings a new perspective in which the "graph topology inference problem" and signal recovery is converted to the "parameter estimation" of a Gaussian distribution and signal denoising from measurements. This point of view shows how the GSP domain problems are close to the classical signal processing ones. In the same way, for K independent

measurements, the distribution can be rewritten as follows

$$\prod_{k=1}^{K} p\big(\mathbf{x}_k \mid \mathbf{\Sigma}^{-1}\big) = (2\pi)^{\frac{-KN}{2}} \mid \mathbf{\Sigma} \mid^{\frac{-K}{2}} \prod_{k=1}^{K} \exp\big(-\frac{1}{2}\mathbf{x}_k^T \mathbf{\Sigma}^{-1}\mathbf{x}_k\big). \tag{3.5}$$

Here, a simple and naive topology estimation is via the maximum likelihood estimation. Since it is a Gaussian distribution which has an exponential term, we take a logarithm first, i.e. the log-likelihood estimation, as follows

$$\Sigma_{k=1}^{K} \log p\big(\mathbf{x}_k \mid \mathbf{\Sigma}^{-1}\big) = -\frac{K}{2}\log \mid \mathbf{\Sigma}^{-1} \mid + \frac{1}{2}\Sigma_{k=1}^{K} \mathrm{Tr}\big(\mathbf{x}_k^T \mathbf{\Sigma}^{-1}\mathbf{x}_k\big) \tag{3.6}$$

where $\Sigma_{k=1}^{K}$ denotes the summation over k. Hence by maximization, we have the following estimation

$$\widehat{\mathbf{\Sigma}} = \min_{\mathbf{\Sigma}} \mathrm{Tr}\big(\mathbf{\Sigma}^{-1}\mathbf{S}\big) - \log \mid \mathbf{\Sigma}^{-1} \mid \tag{3.7}$$

which is the basic optimization problem called inverse covariance estimation, discussed earlier in 3.1. The elements of the inverse covariance estimation matrix may be illustrated by the following relations [81]

$$\mathbb{E}\big[x_i \mid \mathbf{x}_{-i}\big] = -\frac{1}{\big(\mathbf{\Sigma}^{-1}\big)_{ii}} \Sigma_{j\neq i} \big(\mathbf{\Sigma}^{-1}\big)_{ij} x_j, \tag{3.8}$$

where $\mathbf{x}_{-i}$ denotes all the elements of data vector $\mathbf{x}$ except x_i. Also, the partial correlation is computed as follows

$$\mathrm{Corr}\big[x_i x_j \mid \mathbf{x}_{-ij}\big] = -\frac{\big(\mathbf{\Sigma}^{-1}\big)_{ij}}{\sqrt{\big(\mathbf{\Sigma}^{-1}\big)_{ii}\big(\mathbf{\Sigma}^{-1}\big)_{jj}}}, \quad i \neq j, \tag{3.9}$$

where $\mathbf{x}_{-ij}$ denotes all the elements of data vector $\mathbf{x}$ excluding x_i and x_j.

3.2.2 The Main Idea

Assume that the data matrix $\mathbf{Y}$ is given which contains noisy measurements of graph signals in its columns. The size of $\mathbf{Y}$ is $N \times K$ where N is the number of vertices and K is the number of measurements. The k'th measurement is given as follows

$$\mathbf{y}[k] = \mathbf{x}[k] + \mathbf{e}[k], k = 1, \dots K. \tag{3.10}$$

Moreover, we adopt a multivariate Gaussian distribution for the noise $\mathbf{e}[k]$, given

as follows

$$p(\mathbf{e}[k];\sigma_e) \sim \mathcal{N}(\mathbf{0}_N, \sigma_e \mathbf{I}_N) \tag{3.11}$$

To find the graph topology, we utilize factor analysis to represent the signals via a matrix linked to the graph Laplacian matrix directly. Each measurement is modeled by $\mathbf{x}[k] = \boldsymbol{\chi}\mathbf{h}[k]+\mathbf{u}[k],\ \forall k$, where the hidden or latent variable $\mathbf{h}[k] \in \mathbb{R}^N$ relates to the graph signal via the eigenvector matrix $\boldsymbol{\chi}$ [82] and $\mathbf{u}[k] \in \mathbb{R}^N$ is the mean of the graph signal $\mathbf{x}[k]$. It is also assumed that $\forall k, \mathbf{u}[k] = \mathbf{0}_N$. As discussed in [41], by this model, each graph signal contributes to the underlying graph structure estimation. Following [41], it is assumed that $\mathbf{h}[k]$ has a degenerate zero mean multivariate Gaussian distribution as follows[2]

$$p\big(\mathbf{h}[k];\boldsymbol{\Lambda}^\dagger\big) \sim \mathcal{N}(\mathbf{0}_N, \boldsymbol{\Lambda}^\dagger), \tag{3.12}$$

where $\boldsymbol{\Lambda}$ is the precision matrix. The matrix form of the measurements is as follows

$$\mathbf{Y} = \boldsymbol{\chi}\mathbf{H} + \mathbf{E}, \tag{3.13}$$

where $\mathbf{H} = \big[\mathbf{h}[1], \ldots, \mathbf{h}[K]\big]$ and $\mathbf{E} = \big[\mathbf{e}[1], \ldots, \mathbf{e}[K]\big]$. By using the Kronecker product property [3], the vector form of the given data is as follows

$$\mathbf{y} = \mathbf{B}\mathbf{h} + \mathbf{e}, \tag{3.14}$$

where $\mathbf{y} = \text{vec}\big(\mathbf{Y}\big)$, $\mathbf{B} = \mathbf{I}_K \otimes \boldsymbol{\chi}$, $\mathbf{h} = \text{vec}\big(\mathbf{H}\big)$, and $\mathbf{e} = \text{vec}\big(\mathbf{E}\big)$. Thus, we have

$$p\big(\mathbf{h};\boldsymbol{\Lambda}^\dagger\big) \sim \mathcal{N}(\mathbf{0}_N, \mathbf{C}_0^\dagger), \tag{3.15}$$

$$p\big(\mathbf{y} \mid \mathbf{h};\boldsymbol{\chi},\sigma_e\big) \sim \mathcal{N}\big(\mathbf{B}\mathbf{h}, \sigma_e\mathbf{I}_{NK}\big), \tag{3.16}$$

$$p\big(\mathbf{y};\boldsymbol{\Lambda}^\dagger,\boldsymbol{\chi},\sigma_e\big) \sim \mathcal{N}\big(\mathbf{0}, \mathbf{B}\mathbf{C}_0^\dagger\mathbf{B}^T + \sigma_e\mathbf{I}_{NK}\big), \tag{3.17}$$

where $\mathbf{C}_0 = \mathbf{I}_K \otimes \boldsymbol{\Lambda}$. [83] tell us that by using the identity $\boldsymbol{\chi}\boldsymbol{\chi}^T = \mathbf{I}_N$ and the Kronecker product property, we have

$$\begin{aligned}\mathbf{B}\mathbf{C}_0^\dagger\mathbf{B}^T &= \big(\mathbf{I}_K \otimes \boldsymbol{\chi}\big)\big(\mathbf{I}_K \otimes \boldsymbol{\Lambda}^\dagger\big)\big(\mathbf{I}_K \otimes \boldsymbol{\chi}\big)^T \\ &= \big(\mathbf{I}_K \otimes \boldsymbol{\chi}\boldsymbol{\Lambda}^\dagger\boldsymbol{\chi}^T\big),\end{aligned} \tag{3.18}$$

[2] A degenerate distribution is a probability distribution in a space with specific dimension but with a support only on a space of lower dimension. For a simple example, consider there is a dice with three sides of "1" and three sides of "6" and then when it is rolled, either "1" or "6" shows up with the probability of 0.5.

[3] If $\mathbf{A}_1, \mathbf{A}_2$, and $\mathbf{A}_3$ are three matrices with appropriate dimensions, we have $\text{vec}(\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3) = (\mathbf{A}_3 \otimes \mathbf{A}_1)\text{vec}(\mathbf{A}_2)$.

and thus, the covariance matrix of $p(\mathbf{y})$ is as follows

$$\mathbf{B}\mathbf{C}_0^{\dagger}\mathbf{B}^T + \sigma_e\mathbf{I}_{NK} = \Big(\mathbf{I}_K \otimes \big(\mathbf{L}^{\dagger} + \sigma_e\mathbf{I}_N\big)\Big), \tag{3.19}$$

where by using (2.16)

$$\mathbf{L}^{\dagger} = \boldsymbol{\chi}\boldsymbol{\Lambda}^{\dagger}\boldsymbol{\chi}^T. \tag{3.20}$$

By applying the Bayes rule, we have

$$p\big(\mathbf{h} \mid \mathbf{y}; \boldsymbol{\Lambda}^{\dagger}, \boldsymbol{\chi}, \sigma_e\big) = \frac{p\big(\mathbf{y} \mid \mathbf{h}; \boldsymbol{\chi}, \sigma_e\big)p\big(\mathbf{h}; \boldsymbol{\Lambda}^{\dagger}\big)}{p\big(\mathbf{y}; \boldsymbol{\Lambda}^{\dagger}, \boldsymbol{\chi}, \sigma_e\big)}, \tag{3.21}$$

and the MMSE estimator of the latent variable is given as follows

$$\hat{\mathbf{h}} = \mathbb{E}\big[\mathbf{h} \mid \mathbf{y}\big] = \boldsymbol{\mu}_{\mathbf{h}}, \tag{3.22}$$

By substituting (3.15)-(3.17) in (3.21) and some manipulations, [83] tells us that we have a Gaussian posterior distribution function as follows

$$p\big(\mathbf{h} \mid \mathbf{y}; \boldsymbol{\Lambda}^{\dagger}, \boldsymbol{\chi}, \sigma_e\big) \sim \mathcal{N}\big(\boldsymbol{\mu}_{\mathbf{h}}, \mathbf{C}_{\mathbf{h}}\big) \tag{3.23}$$

with the mean vector

$$\boldsymbol{\mu}_{\mathbf{h}} = \frac{1}{\sigma_e}\mathbf{C}_{\mathbf{h}}\mathbf{B}^T\mathbf{y}, \tag{3.24}$$

and the covariance matrix

$$\begin{aligned}\mathbf{C}_{\mathbf{h}} &= \big(\mathbf{C}_0 + \frac{1}{\sigma_e}\mathbf{B}^T\mathbf{B}\big)^{-1} \\ &= \big(\mathbf{C}_0 + \frac{1}{\sigma_e}\mathbf{I}_{NK}\big)^{-1} \\ &= \mathbf{I}_K \otimes \big(\boldsymbol{\Lambda} + \frac{1}{\sigma_e}\mathbf{I}_N\big)^{-1}\end{aligned} \tag{3.25}$$

Since $\mathbf{x}$ has a linear relationship with $\mathbf{h}$, i.e. $\mathbf{x} = \mathbf{B}\mathbf{h}$, we have $\hat{\mathbf{x}} = \mathbf{B}\boldsymbol{\mu}_{\mathbf{h}}$ and

$$\hat{\mathbf{x}} = \Big(\mathbf{I}_K \otimes \big(\mathbf{I}_N + \sigma_e\mathbf{L}\big)^{-1}\Big)\mathbf{y}, \tag{3.26}$$

where σ_e and $\mathbf{L}$ are estimated in the next. Also, it can be represented by the matrix form as follows [4]

$$\hat{\mathbf{X}} = \big(\mathbf{I}_N + \sigma_e\mathbf{L}\big)^{-1}\mathbf{Y} \tag{3.27}$$

where it is similar to the one in [41], if the filter coefficient σ_e is replaced by α (see more details about this optimization problem in (3.82)). In one hand, if $\sigma_e = 0$,

the filtered graph signal is exactly equivalent to the observation since there is no noise corruption. On the other hand, if the variance of the noise distribution is too large, $\mathbf{X} = \mathbf{0}$ which shows that the observations are not important as long as the noise power is large enough. In [41], the coefficient α came from the smoothness prior term of graph signals over the topology which also exists in the minimization problem to find the optimum value of the graph Laplacian as a regularization term. However, α was not investigated analytically in [41] and it was estimated by a grid exhaustive search. Here, to estimate σ_e and $\mathbf{\Lambda}^\dagger$, an expectation maximization (EM) algorithm is implemented which maximizes

$$Q\big(\mathbf{\Lambda}^\dagger, \boldsymbol{\chi}, \sigma_e\big) = \mathbb{E}_{\mathbf{h}|\mathbf{y};\hat{\sigma}_e,\hat{\boldsymbol{\chi}},\hat{\mathbf{\Lambda}}^\dagger}\big[\log p\big(\mathbf{y}, \mathbf{h}; \sigma_e, \boldsymbol{\chi}, \mathbf{\Lambda}^\dagger\big)\big], \tag{3.28}$$

where $\hat{\mathbf{\Lambda}}^\dagger$, $\hat{\boldsymbol{\chi}}$, and $\hat{\sigma}_e$ are the estimations of $\mathbf{\Lambda}^\dagger$, $\boldsymbol{\chi}$, and σ_e in the previous iteration. Hereafter, according to [4], to simplify the notation, we define $\mathbf{\Xi} := \big(\sigma_e, \boldsymbol{\chi}, \mathbf{\Lambda}^\dagger\big)$. The function Q in (3.28) can be separated as follows

$$Q(\mathbf{\Xi}) = Q_1\big(\boldsymbol{\chi}, \sigma_e\big) + Q_2\big(\mathbf{\Lambda}^\dagger\big), \tag{3.29}$$

$$Q_1\big(\boldsymbol{\chi}, \sigma_e\big) = \mathbb{E}_{\mathbf{h}|\mathbf{y};\hat{\mathbf{\Xi}}}\big[\log p\big(\mathbf{y} \mid \mathbf{h}; \boldsymbol{\chi}, \sigma_e\big)\big], \tag{3.30}$$

$$Q_2\big(\mathbf{\Lambda}^\dagger\big) = \mathbb{E}_{\mathbf{h}|\mathbf{y};\hat{\mathbf{\Xi}}}\big[\log p\big(\mathbf{h}; \mathbf{\Lambda}^\dagger\big)\big]. \tag{3.31}$$

In mathematics, an EM algorithm is an iterative approach to determine the maximum a posteriori estimates of parameters where the model depends on the latent variable. This algorithm alternates between the expectation step (E-step) and the maximization step (M-step). The former computing the expected value of the log-likelihood of the current estimate for the parameters, and the latter maximizes the former. These estimates are finally applied to find the distribution of the latent variables in the next iteration of the E-step. In the same way, find the optimum parameters here, Q must be maximized for each argument iteratively. To maximize with respect to σ_e, the Q_2 is not relevant and thus Q_1 is simplified as follows[4]

$$\begin{aligned} Q_1(\boldsymbol{\chi}, \sigma_e) =& \mathbb{E}_{\mathbf{h}|\mathbf{y};\hat{\mathbf{\Xi}}}\big[\log p\big(\mathbf{y} \mid \mathbf{h}; \boldsymbol{\chi}, \sigma_e\big)\big] \\ =& -\frac{1}{2}\log 2\pi - \frac{NK}{2}\log \sigma_e - \frac{1}{2\sigma_e}\mathbb{E}_{\mathbf{h}|\mathbf{y};\hat{\mathbf{\Xi}}}\big[\,\|\mathbf{y} - \mathbf{B}\mathbf{h}\|_2^2\,\big] \\ \overset{(a)}{\propto}& -\frac{NK}{2}\log \sigma_e - \frac{1}{2\sigma_e}\mathrm{Tr}\big(\mathbf{C}_\mathbf{h}\big), \end{aligned} \tag{3.32}$$

[4]the $\propto$ notation indicates that if a term is not contributing to the σ_e optimization, it is dropped. In other words, the expression on the right side of $\propto$ is proportional to the one on the left side, and thus maximization over one expression is equal to the maximization over the other expression.

where $\overset{(a)}{=}$ is valid since the first term is constant in the maximization problem. Setting the derivative of (3.32) with respect to σ_e equal to zero, we adopted the following equation from [4]

$$-\frac{N}{\sigma_e}+\text{Tr}\Big(\big(\sigma_e\mathbf{L}+\mathbf{I}\big)^{-1}\cdot\mathbf{L}\cdot\big(\sigma_e\mathbf{L}+\mathbf{I}\big)^{-1}\Big)=0, \tag{3.33}$$

and then σ_e is found by a numerical algorithm.

In the second step of EM procedure, we should maximize $Q_1(\cdot)$ with respect to $\boldsymbol{\chi}$. Since we are not interested in the eigenvector matrix and the goal is to estimate the Laplacian matrix, we try to change the variable in Q_2. By using $\mathbf{L}=\boldsymbol{\chi}\boldsymbol{\Lambda}\boldsymbol{\chi}^T$ and applying the eigenvector matrix identity $\boldsymbol{\chi}\boldsymbol{\chi}^T=\boldsymbol{\chi}^T\boldsymbol{\chi}=\mathbf{I}_N$ and maximizing $Q_2(\boldsymbol{\Lambda})$ with respect to $\boldsymbol{\Lambda}$, the Laplacian matrix is estimated. According to [4], (3.15) is used to find Q_2 as follows

$$\begin{aligned}Q_2\big(\boldsymbol{\Lambda}^\dagger\big)&\propto\mathbb{E}_{\mathbf{h}|\mathbf{y};\hat{\boldsymbol{\Xi}}}\big[\log|\mathbf{C}_0|-\mathbf{h}^T\mathbf{C}_0\mathbf{h}\big]\\&=K\cdot\log|\boldsymbol{\Lambda}|-\mathbb{E}_{\mathbf{h}|\mathbf{y};\hat{\boldsymbol{\Xi}}}\big[\mathbf{x}^T\big(\mathbf{I}_K\otimes\mathbf{L}\big)\mathbf{x}\big]\\&=K\cdot\log|\mathbf{L}|-\text{Tr}\big(\hat{\mathbf{X}}^T\mathbf{L}\hat{\mathbf{X}}\big)\\&=K\cdot\log|\mathbf{L}|-K\cdot\text{Tr}\big(\mathbf{S}\mathbf{L}\big),\end{aligned} \tag{3.34}$$

where $\mathbf{S}=\frac{1}{K}\hat{\mathbf{X}}\hat{\mathbf{X}}^T$ is the empirical covariance matrix of measurements and $|\cdot|$ notation is used to denote the pseudo-determinant due to the singularity of $\boldsymbol{\Lambda}$ and $\mathbf{C}_0$. As a simple definition, the pseudo-determinant is the product of all non-zero eigenvalues. Then, the last step of EM algorithm is to maximize (3.29) with respect to the Laplacian matrix as follows [83]

$$\begin{aligned}\underset{\mathbf{L}}{\text{argmax}}\ \ &\log|\mathbf{L}|-\text{Tr}\big(\mathbf{S}\mathbf{L}\big)-\text{Tr}\big((\sigma_e\mathbf{L}+\mathbf{I})^{-1}\big)\\\text{s.t.}\ \ &\mathbf{L}_{ij}=\mathbf{L}_{ji},\\&\mathbf{L}_{ij}\leq 0\ \text{if}\ i\neq j,\\&\mathbf{L}\cdot\mathbf{1}_N=\mathbf{0}_N,\\&\text{Tr}(\mathbf{L})=c,\end{aligned} \tag{3.35}$$

where the first three constraints guarantee a valid Laplacian, according to the definition in Sec. 2.1.3.4. The last constraint avoids the trivial solution. These constraints are applied for most of the graph Laplacian inference problem. The first and third terms in (3.35) makes it difficult to be solved. To relax the third term and have a lower computational complexity problem, the upper and lower bounds for the ma-

trix inverse term are computed by using the Theorem 1 of [84] as follows

$$\mathrm{Tr}\Big(\big(\sigma_e\mathbf{L}+\mathbf{I}\big)^{-1}\Big) \leq N - \frac{c^2\sigma_e}{\sigma_e\left\|\mathbf{L}\right\|_F^2 - c}, \tag{3.36}$$

$$\mathrm{Tr}\Big(\big(\sigma_e\mathbf{L}+\mathbf{I}\big)^{-1}\Big) \geq \frac{N}{2\sigma_e+1} - \frac{\frac{c^2\sigma_e-2cN\sigma_e+2N^2\sigma_e}{2\sigma_e+1}}{\sigma_e\left\|\mathbf{L}\right\|_F^2 - c - 2N - 2c\sigma_e}, \tag{3.37}$$

and for a constant σ_e, minimizing $\sigma_e\left\|\mathbf{L}\right\|_F^2$ leads to minimization of $\mathrm{Tr}\Big(\big(\sigma_e\mathbf{L}+\mathbf{I}\big)^{-1}\Big)$. To prevent the trivial solution and control the sparsity, the following optimization problem has been proposed in [4]

$$\begin{aligned}
\underset{\mathbf{L}}{\text{argmin}} \quad & -\text{logdet}\left|\mathbf{L}\right| + \mathrm{Tr}\big(\mathbf{SL}\big) + \sigma_e\left\|\mathbf{L}\right\|_F^2 \\
\text{s.t.} \quad & \mathbf{L}_{ij} = \mathbf{L}_{ji}, \\
& \mathbf{L}_{ij} \leq 0 \text{ if } i \neq j, \\
& \mathbf{L}\cdot\mathbf{1}_N = \mathbf{0}_N, \\
& \mathrm{Tr}(\mathbf{L}) = c.
\end{aligned} \tag{3.38}$$

To solve the pseudo determinant term issue, a change of variable is applied. We replace $\mathbf{L}$ by $\mathbf{L}+\mathbf{J}$, where $\mathbf{J} = \frac{1}{N}\mathbf{1}_N\mathbf{1}_N^T$ to remove the effect of zero eigenvalue and hence singularity of the Laplacian matrix. Finally, the optimization problem is rewritten as follows

$$\begin{aligned}
\underset{\mathbf{L}}{\text{argmin}} \quad & -\text{logdet}\left(\mathbf{L}+\mathbf{J}\right) + \mathrm{Tr}\big(\mathbf{SL}\big) + \sigma_e\left\|\mathbf{L}\right\|_F^2 \\
\text{s.t.} \quad & \mathbf{L}_{ij} = \mathbf{L}_{ji}, \\
& \mathbf{L}_{ij} \leq 0 \text{ if } i \neq j, \\
& \mathbf{L}\cdot\mathbf{1}_N = \mathbf{0}_N, \\
& \mathrm{Tr}(\mathbf{L}) = c.
\end{aligned} \tag{3.39}$$

Since (3.39) is convex, any available solver can be used, e.g. YALMIP [85]. In the next, we propose an efficient solver via the proximal point algorithm. To sum up, our proposed method called Bayesian Topology Learning (BTL), iterates between three steps as shown in Algorithm 3.1.

Algorithm 3.1: Bayesian Topology Learning (BTL), taken from [83].

Input: The Measurements $\mathbf{Y}$
Initialize: $\hat{\mathbf{X}} = \mathbf{Y}$, $\hat{\sigma}_e = \sigma_0$, and c
while *Not Converged* **do**
  Laplacian matrix learning by solving (3.39),
  Noise variance estimation via (3.33),
  Signal recovery: $\hat{\mathbf{X}} = \left(\mathbf{I}_N + \hat{\sigma}_e \hat{\mathbf{L}}\right)^{-1} \mathbf{Y}$.
end
Output: $\hat{\mathbf{L}}$ and $\hat{\mathbf{X}}$.

3.3 Bayesian Topology Learning via Proximal Point Algorithm

A fast algorithm to solve (3.39) is investigated. Using Proposition 2, the last three constraints are reshaped in the form of the inner product of two matrices, resulting in a simpler Lagrangian function. Then, the Lagrangian function is respectively maximized and minimized with respect to the dual and primal variables (graph Laplacian matrix). A proximal point algorithm is proposed for the minimization step. The maximization is done via the Newton or quasi-Newton methods. Theorem 1 computes a condition guaranteeing the convergence of iteration between the maximization and minimization problems.

Proposition 3.3.1. *[4]: The Laplacian matrix is a symmetric matrix and hence the summation over all entries of the i'th row (or the i'th column) is rewritten as follows*

$$\sum_{j=1}^{N} L_{ij} = \frac{1}{2} Tr\Big(\mathbf{L}\big(\mathbf{U}_i + \mathbf{U}_i^T\big)\Big) \tag{3.40}$$

where $\mathbf{U}_i$ is an $N \times N$ zero matrix in which the i'th column is only ones.

Proposition 3.3.2. *[4]: The constraint $\mathbf{L}_{ij} \leq 0$ if $i \neq j$ in the problem (3.39) can be replaced by $\|\mathbf{L}\|_1 = 2 \cdot Tr(\mathbf{L})$, since $\mathbf{L}$ is a positive semi-definite matrix and $\mathbf{L} \cdot \mathbf{1} = \mathbf{0}$.*

Proof. Since there is no self-loop, the graph weight matrix's diagonal has zero en-

tries. Having $\mathbf{L}_{ij} \leq 0$ if $i \neq j$ is equivalent to $\|\mathbf{D}\|_1 = \|\mathbf{W}\|_1$ and thus

$$\begin{aligned} \|\mathbf{L}\|_1 &= \|\mathbf{D} - \mathbf{W}\|_1 \\ &= \|\mathbf{D}\|_1 + \|\mathbf{W}\|_1 \\ &= 2\,\|\mathbf{D}\|_1 \\ &= 2 \cdot \text{Tr}(\mathbf{D}) \\ &= 2 \cdot \text{Tr}(\mathbf{L}) \end{aligned} \tag{3.41}$$

Here, the graph Laplacian matrix is a positive semi-definite matrix ensuring $\mathbf{L}_{ii} \geq 0$ so as $\mathbf{L}_{ij} \leq 0$ if $i \neq j$. □

By some simple manipulations, the minimization problem of (3.39) can be rewritten in general form as follows[5]

$$\begin{aligned} \underset{\mathbf{L}}{\text{argmin}} \quad & -\text{logdet}\,(\mathbf{L} + \mathbf{J}) + \text{Tr}\big(\mathbf{L}\mathbf{S}\big) + \sigma_e\,\|\mathbf{L}\|_F^2 \\ \text{s.t.} \quad & \mathcal{B}(\mathbf{L}) = \mathbf{a}, \end{aligned} \tag{3.42}$$

where $\mathbf{a} = [\mathbf{0};\, c;\, 2c/N]^T$, and $\mathcal{B}(\cdot) : \mathcal{R}^{N\times N} \rightarrow \mathcal{R}^{(N+2)\times 1}$ is an operator defined as follows [4]

$$\mathcal{B}(\mathbf{L}) = \left[\frac{1}{2}\text{Tr}\Big(\mathbf{L}\big(\mathbf{U}_1 + \mathbf{U}_1^T\big)\Big), \ldots, \frac{1}{2}\text{Tr}\Big(\mathbf{L}\big(\mathbf{U}_N + \mathbf{U}_N^T\big)\Big), \text{Tr}(\mathbf{L}),\, \|\mathbf{L}\|_1\right]^T, \tag{3.43}$$

We assume that the feasible set $\mathcal{F} = \{\mathbf{L} \in \mathcal{S}_+^N,\ \mathcal{B}(\mathbf{L}) = \mathbf{a}\}$ is not an empty set and since (3.42) is convex, there exists a global solution. The Lagrangian function for the primal variable $\mathbf{L}$ and the dual variable $\boldsymbol{\nu} = [\nu_1, \ldots, \nu_N, \nu_{N+1}, \nu_{N+2}]$ is as follows

$$\mathcal{L}\big(\mathbf{L}; \boldsymbol{\nu}\big) = -\text{logdet}\,(\mathbf{L} + \mathbf{J}) + \text{Tr}\big(\mathbf{L}\mathbf{S}\big) + \sigma_e\,\|\mathbf{L}\|_F^2 + \boldsymbol{\nu}^T(\mathbf{a} - \mathcal{B}(\mathbf{L})), \tag{3.44}$$

and $\mathbf{L}$ can be estimated via the following optimization problem

$$\hat{\mathbf{L}} = \underset{\mathbf{L}}{\text{argmin}}\ g(\mathbf{L}), \tag{3.45}$$

where

$$g(\mathbf{L}) = \max_{\boldsymbol{\nu}}\, \mathcal{L}\big(\mathbf{L}; \boldsymbol{\nu}\big). \tag{3.46}$$

Since it is difficult to implement (3.45) directly, the proximal algorithm is pro-

[5]This minimization problem is reduced to the one proposed in [41] for $\alpha_1 = 0$.

posed in [4], which is a standard tool to solve a constrained and nonsmooth optimization problem [86]. The *proximal operator* of a closed proper convex function $f : \mathcal{R}^n \to \mathcal{R} \cup \{+\infty\}$ is denoted $\mathbf{prox}_f : \mathcal{R}^n \to \mathcal{R}^n$ and defined as follows

$$\mathbf{prox}_{\eta f}(\mathbf{v}) = \underset{\mathbf{y}}{\operatorname{argmin}} \ \big(f(\mathbf{y}) + \frac{1}{2\eta}\|\mathbf{y}-\mathbf{v}\|_2^2\big). \tag{3.47}$$

where η is a scale parameter. To solve the optimization problem $\hat{\mathbf{y}} = \underset{\mathbf{y}}{\operatorname{argmin}} \ f(\mathbf{y})$, the *proximal point algorithm*, also called proximal iteration iterates as follows

$$\mathbf{y}^{(t+1)} := \mathbf{prox}_{\eta f}\big(\mathbf{y}^{(t)}\big) \tag{3.48}$$

until the error between two consecutive $\mathbf{y}$ is less than a threshold. By some iterations over t, $\mathbf{y}^t$ and $f\big(\mathbf{y}^t\big)$ converge to the minimum of f and its optimal value [87]. The proximal operator makes a smooth version of the function using the regularization. Another approach to compute the proximal operator is as follows [86]

$$\mathbf{prox}_{\eta f}\big(\mathbf{y}\big) = \mathbf{y} - \eta \nabla M_{\eta f}(\mathbf{y}), \tag{3.49}$$

where ∇ and $M_{\eta f}(\mathbf{y})$ are the gradient operator and the Moreau-Yosida regularization, respectively. The Moreau-Yosida regularization with parameter $\eta > 0$ is defined as follows [86, 88, 89]

$$\begin{aligned} M_{\eta g}(\mathbf{L}) &= \min_{\mathbf{L}'} \ \big\{ g\big(\mathbf{L}'\big) + \frac{1}{2\eta}\|\mathbf{L}-\mathbf{L}'\|_F^2 \big\} \\ &\overset{(a)}{=} \max_{\boldsymbol{\nu}} \ \min_{\mathbf{L}'} \big\{ \mathcal{L}\big(\mathbf{L}';\boldsymbol{\nu}\big) + \frac{1}{2\eta}\|\mathbf{L}-\mathbf{L}'\|_F^2 \big\} \\ &= \max_{\boldsymbol{\nu}} \ \mathbf{P}_\eta\big(\mathbf{L};\boldsymbol{\nu}\big), \end{aligned} \tag{3.50}$$

where $\overset{(a)}{=}$ is based on the Von Neumann-Fan minimax theorem. Similar to the proximal operator, the Moreau-Yosida regularization generates a smooth version of the function. Besides, $M_{\eta g}(\mathbf{L})$ and $g\big(\mathbf{L}\big)$ have the same minimizers and thus we are after (3.50) instead of (3.45), equivalently. All things considered, we try to minimize $M_{\eta g}(\mathbf{L})$ (instead of $g(\mathbf{L})$), by implementing the proximal point algorithm proposed in (3.48).

3.3.1 Derivation of $\mathbf{P}_\eta\big(\mathbf{L};\boldsymbol{\nu}\big)$

The operator $\mathcal{B}_a$ which is the adjoint of $\mathcal{B}$ is computed as follows

$$\begin{aligned}\mathcal{B}_a(\boldsymbol{\nu}) &= \frac{\partial\langle \mathbf{L}, \mathcal{B}_a(\boldsymbol{\nu})\rangle}{\partial \mathbf{L}} \\ &\overset{(a)}{=} \frac{\partial\langle \mathcal{B}(\mathbf{L}), \boldsymbol{\nu}\rangle}{\partial \mathbf{L}},\end{aligned} \tag{3.51}$$

where $\frac{\partial}{\partial \mathbf{L}}$ denotes the partial derivative and (a) uses the inner product property. By applying (3.43) and some manipulations,

$$\mathcal{B}_a(\boldsymbol{\nu}) = \frac{1}{2}\nu_1\big(\mathbf{U}_1+\mathbf{U}_1^T\big)+\cdots+\frac{1}{2}\nu_N\big(\mathbf{U}_N+\mathbf{U}_N^T\big)+\nu_{N+1}\mathbf{I}+\nu_{N+2}\big(2\mathbf{I}-N\mathbf{J}\big). \tag{3.52}$$

We introduce $\mathbf{T}_\eta$ as

$$\mathbf{T}_\eta\big(\mathbf{L};\boldsymbol{\nu}\big) \triangleq \frac{1}{1+2\eta\sigma_e}\Big(\mathbf{L}-\eta\big(\mathbf{S}-\mathcal{B}_a(\boldsymbol{\nu})\big)\Big), \tag{3.53}$$

and $\mathbf{P}_\eta\big(\mathbf{L};\boldsymbol{\nu}\big)$ is therefor given as below

$$\mathbf{P}_\eta\big(\mathbf{L};\boldsymbol{\nu}\big) = \boldsymbol{\nu}^T\mathbf{a}+\frac{1}{2\eta}\left\|\mathbf{L}\right\|_F^2-\frac{1+2\eta\sigma_e}{2\eta}\left\|\mathbf{T}_\eta\big(\mathbf{L};\boldsymbol{\nu}\big)\right\|_F^2+\min_{\mathbf{L}'}\mathcal{J}_\eta\big(\mathbf{L}',\mathbf{L};\boldsymbol{\nu}\big) \tag{3.54}$$

and

$$\mathcal{J}_\eta\big(\mathbf{L}',\mathbf{L};\boldsymbol{\nu}\big) = -\text{logdet}\left(\mathbf{L}'\right)+\frac{1+2\eta\sigma_e}{2\eta}\left\|\mathbf{L}'-\mathbf{T}_\eta\big(\mathbf{L};\boldsymbol{\nu}\big)\right\|_F^2. \tag{3.55}$$

The following Lemma helps finding the minimizer of $\mathcal{J}_\eta\big(\mathbf{L}',\mathbf{L};\boldsymbol{\nu}\big)$ and simplifying (3.54).

Lemma 3.3.3. *[90]: Let us assume $\mathbf{Z}\in\mathcal{S}^N$ has the eigenvalue decomposition $\mathbf{Z}=\mathbf{P}\boldsymbol{\Psi}\mathbf{P}^T$ and $\gamma>0$ are given, and $\boldsymbol{\Psi}=diag(\boldsymbol{\psi})$ is the eigenvalue matrix. Assume there are two functions $\phi_\gamma^+(x)\triangleq\frac{1}{2}\big(\sqrt{x^2+4\gamma}+x\big)$ and $\phi_\gamma^-(x)\triangleq\frac{1}{2}\big(\sqrt{x^2+4\gamma}-x\big)$ for all $x\in\mathbb{R}$ and $\mathbf{Z}_1,\mathbf{Z}_2$ are defined as follows*

$$\begin{aligned}\mathbf{Z}_1 &:= \phi_\gamma^+(\mathbf{Z}) = \mathbf{P}diag(\phi_\gamma^+(\boldsymbol{\psi}))\mathbf{P}^T \\ \mathbf{Z}_2 &:= \phi_\gamma^-(\mathbf{Z}) = \mathbf{P}diag(\phi_\gamma^-(\boldsymbol{\psi}))\mathbf{P}^T\end{aligned} \tag{3.56}$$

Then,

1. *$\mathbf{Z}=\mathbf{Z}_1-\mathbf{Z}_2$ and $\mathbf{Z}_1\cdot\mathbf{Z}_2=\gamma\mathbf{I}_N$,*
2. *ϕ_γ^+ is continuously differentiable and its derivative for every $\mathbf{H}\in\mathcal{S}^N$ is given*

as below

$$\left.\frac{\partial \phi_{\gamma}^{+}}{\partial x}\right|_{x=\mathbf{Z}} \cdot [\mathbf{H}] = \mathbf{P}\big(\mathbf{\Omega} \odot (\mathbf{P}^T \mathbf{H} \mathbf{P})\big)\mathbf{P}^T, \tag{3.57}$$

where $\odot$ is the Hadamard product notation and $\mathbf{\Omega}$ is a symmetric matrix defined as follows

$$\Omega_{ij} = \frac{\phi_{\gamma}^{+}(\Psi_i) + \phi_{\gamma}^{+}(\Psi_j)}{\sqrt{\Psi_i^2 + 4\gamma} + \sqrt{\Psi_j^2 + 4\gamma}},\ 1 < i, j < N. \tag{3.58}$$

3.

$$\left.\frac{\partial \phi_{\gamma}^{+}}{\partial x}\right|_{x=\mathbf{Z}} \cdot [\mathbf{Z}_1 + \mathbf{Z}_2] = \phi_{\gamma}^{+}(\mathbf{Z}). \tag{3.59}$$

The first derivative of $\mathcal{J}_{\eta}\big(\mathbf{L}', \mathbf{L}; \boldsymbol{\nu}\big)$ with respect to $\mathbf{L}'$ must be set to zero as given below

$$-\big(\mathbf{L}'\big)^{-1} + \frac{1 + 2\eta\sigma_e}{\eta}\big(\mathbf{L}' - \mathbf{T}_{\eta}\big(\mathbf{L}; \boldsymbol{\nu}\big)\big) = \mathbf{0}_N, \tag{3.60}$$

and then we have

$$\mathbf{L}' = \phi_{\gamma'}^{+}\big(\mathbf{T}_{\eta}(\mathbf{L}; \boldsymbol{\nu})\big), \tag{3.61}$$

where $\gamma' = \frac{\eta}{1+2\eta\sigma_e}$. This $\mathbf{L}'$ minimizes $\mathcal{J}_{\eta}\big(\mathbf{L}', \mathbf{L}; \boldsymbol{\nu}\big)$ and then (3.54) is simplified as follows

$$\begin{aligned}
\mathbf{P}_{\eta}\big(\mathbf{L}; \boldsymbol{\nu}\big) &= \boldsymbol{\nu}^T\mathbf{a} + \frac{1}{2\eta}\left\|\mathbf{L}\right\|_F^2 - \frac{1}{2\gamma'}\left\|\mathbf{T}_{\eta}\big(\mathbf{L}; \boldsymbol{\nu}\big)\right\|_F^2 \\
&\quad - \text{logdet}\left(\phi_{\gamma'}^{+}\big(\mathbf{T}_{\eta}(\mathbf{L}; \boldsymbol{\nu})\big)\right) + \frac{1}{2\gamma'}\left\|\phi_{\gamma'}^{-}\big(\mathbf{T}_{\eta}(\mathbf{L}; \boldsymbol{\nu})\big)\right\|_F^2 \\
&= \boldsymbol{\nu}^T\mathbf{a} + \frac{1}{2\eta}\left\|\mathbf{L}\right\|_F^2 - \text{logdet}\left(\phi_{\gamma'}^{+}\big(\mathbf{T}_{\eta}(\mathbf{L}; \boldsymbol{\nu})\big)\right) + \frac{1}{2\gamma'}\left\|\phi_{\gamma'}^{-}\big(\mathbf{T}_{\eta}(\mathbf{L}; \boldsymbol{\nu})\big)\right\|_F^2 \\
&\quad - \frac{1}{2\gamma'}\left\|\mathbf{T}_{\eta}\big(\mathbf{L}; \boldsymbol{\nu}\big)\right\|_F^2 \\
&\overset{(a)}{=} \boldsymbol{\nu}^T\mathbf{a} + \frac{1}{2\eta}\left\|\mathbf{L}\right\|_F^2 - \text{logdet}\left(\phi_{\gamma'}^{+}\big(\mathbf{T}_{\eta}(\mathbf{L}; \boldsymbol{\nu})\big)\right) \\
&\quad - \frac{1}{2\gamma'}\left\|\phi_{\gamma'}^{+}\big(\mathbf{T}_{\eta}(\mathbf{L}; \boldsymbol{\nu})\big)\right\|_F^2 + N,
\end{aligned} \tag{3.62}$$

where $\overset{(a)}{=}$ follows Lemma 3.3.3.

3.3.2 Derivation of $M_{\eta g}(\mathbf{L})$

Lemma 3.3.4. *[4]: Let us define the following three notations*

$$\begin{aligned} \boldsymbol{\Phi}^{+} &:= \phi^{+}_{\gamma'}\big(\mathbf{T}_{\eta}(\mathbf{L};\boldsymbol{\nu})\big), \\ \boldsymbol{\Phi}^{-} &:= \phi^{-}_{\gamma'}\big(\mathbf{T}_{\eta}(\mathbf{L};\boldsymbol{\nu})\big), \\ \big(\boldsymbol{\Phi}^{+}\big)' &:= \frac{\partial \phi^{+}_{\gamma'}}{\partial \mathbf{T}_{\eta}} \end{aligned} \tag{3.63}$$

Then, the derivative of $\mathbf{P}_{\eta}\big(\mathbf{L};\boldsymbol{\nu}\big)$ *with respect to* $\boldsymbol{\nu}$ *is as follows*

$$\nabla_{\boldsymbol{\nu}}\mathbf{P}_{\eta}\big(\mathbf{L};\boldsymbol{\nu}\big) = \mathbf{a} - \mathcal{B}\Big(\boldsymbol{\Phi}^{+}\Big). \tag{3.64}$$

Proof.

$$\begin{aligned} \nabla_{\boldsymbol{\nu}}\mathbf{P}_{\eta}\big(\mathbf{L};\boldsymbol{\nu}\big) &= \mathbf{a} - \gamma'\mathcal{B}\Big(\big(\boldsymbol{\Phi}^{+}\big)'\big(\boldsymbol{\Phi}^{+}\big)^{-1}\Big) - \mathcal{B}\Big(\big(\boldsymbol{\Phi}^{+}\big)'\boldsymbol{\Phi}^{+}\Big) \\ &= \mathbf{a} - \mathcal{B}\Big(\big(\boldsymbol{\Phi}^{+}\big)'\big[\boldsymbol{\Phi}^{-} + \boldsymbol{\Phi}^{+}\big]\Big) \\ &\overset{(a)}{=} \mathbf{a} - \mathcal{B}\Big(\boldsymbol{\Phi}^{+}\Big) \end{aligned} \quad , \tag{3.65}$$

where $\overset{(a)}{=}$ follows from Lemma 3.3.3, part 3. □

By applying (3.57), we have readily the following corollary.

Corollary 3.3.4.1. *[4]: The Hessian of* $\mathbf{P}_{\eta}\big(\mathbf{L};\boldsymbol{\nu}\big)$ *with respect to* $\boldsymbol{\nu}$ *is readily available from 3.3.4 as follows*

$$\begin{aligned} \nabla^{2}_{\boldsymbol{\nu}\boldsymbol{\nu}}\mathbf{P}_{\eta} = -\gamma'\Big[&\mathcal{B}\Big(\big(\boldsymbol{\Phi}^{+}\big)'\cdot[\frac{1}{2}\big(\mathbf{U}_1 + \mathbf{U}_1^{T}\big)]\Big), \ldots, \mathcal{B}\Big(\big(\boldsymbol{\Phi}^{+}\big)'\cdot[\frac{1}{2}\big(\mathbf{U}_N + \mathbf{U}_N^{T}\big)]\Big), \\ &\mathcal{B}\Big(\big(\boldsymbol{\Phi}^{+}\big)'\cdot[\mathbf{I}]\Big), \mathcal{B}\Big(\big(\boldsymbol{\Phi}^{+}\big)'\cdot[2\mathbf{I} - \mathbf{1}\mathbf{1}^{T}]\Big)\Big], \end{aligned} \tag{3.66}$$

Now, the unconstrained maximization problem of (3.50) is solved by a quasi-Newton method, e.g. L-BFGS. Hereafter, it is assumed that

$$\boldsymbol{\nu}_{\text{opt}} = \underset{\boldsymbol{\nu}}{\operatorname{argmax}}\ \mathbf{P}_{\eta}\big(\mathbf{L};\boldsymbol{\nu}\big), \tag{3.67}$$

and by using (3.50), we have

$$M_{\eta g}(\mathbf{L}) = \mathbf{P}_{\eta}\big(\mathbf{L};\boldsymbol{\nu}_{\text{opt}}\big). \tag{3.68}$$

Algorithm 3.2: Bayesian Topology Learning Using Proximal Point Algorithm (BTL-PPA), taken from [4]

Input: $\hat{\mathbf{X}}$, $\mathbf{a}$, and the error threshold $e_{\text{thr}} > 0$
Initialize: $\mathbf{L}^{(0)} \in \mathcal{S}_+^N$ and $\boldsymbol{\nu}^{(0)} = \mathbf{1}$

while $\left\|\mathbf{L}^{(t+1)} - \mathbf{L}^{(t)}\right\|_F^2 / \eta_t \geq e_{thr}$ **do**

1: $\boldsymbol{\nu}_{\text{opt}}^{(t+1)} = \underset{\boldsymbol{\nu}}{\text{argmax}}\ \mathbf{P}_\eta\big(\mathbf{L}^{(t)}; \boldsymbol{\nu}\big)$,

2: $\mathbf{L}^{(t+1)} = \phi_{\gamma'}^+\big(\mathbf{T}_\eta(\mathbf{L}^{(t)}; \boldsymbol{\nu}_{\text{opt}}^{(t+1)})\big)$,

end
Output: The graph Laplacian matrix $\mathbf{L}$.

Lemma 3.3.5. *[4]: The derivative of $M_{\eta g}(\mathbf{L})$ with respect to $\mathbf{L}$ is as follows*

$$\nabla_{\mathbf{L}} M_{\eta g}(\mathbf{L}) = \frac{1}{\eta}\Big(\mathbf{L} - \phi_{\gamma'}^+\big(\mathbf{T}_\eta(\mathbf{L}; \boldsymbol{\nu}_{opt})\big)\Big). \tag{3.69}$$

Proof. We take the derivative of (3.62) with respect to the graph Laplacian matrix

$$\begin{aligned}
\nabla_{\mathbf{L}} M_{\eta g}(\mathbf{L}) &= \frac{1}{\eta}\mathbf{L} - \frac{1}{1+2\eta\sigma_e}\Big(\big(\boldsymbol{\Phi}^+\big)'\big(\boldsymbol{\Phi}^+\big)^{-1}\Big) - \frac{1}{\gamma'(1+2\eta\sigma_e)}\Big(\big(\boldsymbol{\Phi}^+\big)'\boldsymbol{\Phi}^+\Big) \\
&= \frac{1}{\eta}\Big(\mathbf{L} - \gamma'\big(\boldsymbol{\Phi}^+\big)'\big(\boldsymbol{\Phi}^+\big)^{-1} - \big(\boldsymbol{\Phi}^+\big)'\boldsymbol{\Phi}^+\Big) \\
&= \frac{1}{\eta}\Big(\mathbf{L} - \big(\boldsymbol{\Phi}^+\big)'\big(\boldsymbol{\Phi}^- + \boldsymbol{\Phi}^+\big)\Big) \\
&\overset{(a)}{=} \frac{1}{\eta}\Big(\mathbf{L} - \phi_{\gamma'}^+\big(\mathbf{T}_\eta(\mathbf{L}; \boldsymbol{\nu}_{\text{opt}})\big)\Big)
\end{aligned} \tag{3.70}$$

where (a) follows (3.59). □

Substituting (3.69) in (3.49), the graph Laplacian is updated iteratively as follows

$$\mathbf{L}^{(t+1)} = \phi_{\gamma'}^+\big(\mathbf{T}_\eta(\mathbf{L}^{(t)}; \boldsymbol{\nu}_{\text{opt}}^{(t+1)})\big), \tag{3.71}$$

All steps to update the Laplacian matrix are summarized in Algorithm 3.2. We used η_t to adjust with the iteration index and have a faster convergence. To find η_t, one way is to use the backtracking line search [86, 91].

3.4 Convergence Analysis

Since in this section, the concept of Lipschitz continuity is used, we review it first. In simple words, a Lipschitz continuous function is constrained in the speed of changing with respect to the change of its variable(s). For a Lipschitz continuous function, there is a real number, called the Lipschitz constant, such that the absolute value of the slop of any lines connecting two points on the function's graph is less than or equal to this constant. To define it mathematically, let the function $f : \mathcal{X} \rightarrow \mathcal{Y}$ be defined on the domain $\mathcal{X}$ and the range set $\mathcal{Y}$. If we have two metrics for these metric spaces, respectively as $d_{\mathcal{X}}$ and $d_{\mathcal{Y}}$, to hold Lipschitz continuity

$$d_{\mathcal{Y}}\big(f(x), f(x')\big) \leq l_c \cdot d_{\mathcal{X}}(x, x'), \tag{3.72}$$

for two points x and x' in the domain set of the function and a Lipschitz constant l_c. The smallest possible constant is also called the "best Lipschitz constant". Since in this thesis the real-valued functions is of interest, the Lipschitz continuous function can also be defined simply as follows

$$\mid f(x) - f(x') \mid \leq l_c \mid x - x' \mid \tag{3.73}$$

and thus for the real-valued functions, the Lipschitz constant can be computed simply by finding the upper bound of all the possible slops.

Theorem 3.4.1. *[92]: If f is a convex function on the open convex set ι in a normed linear space and bounded above of one point of ι, it is Lipschitz on any compact subset of ι.*

Theorem 3.4.2. *[92]: If f is a convex function on the open convex set ι for $\mid f \mid \leq z$ in a normed linear space and ι includes a ϵ_0-neighborhood of a subset ι_1, it is Lipchitz on ι_1 with the constant $\frac{2z}{\epsilon_0}$.*

To start discussing the convergence of the proposed algorithm in the previous section, we represent the eigendecomposition of $\mathbf{T}_\eta\big(\mathbf{L}; \boldsymbol{\nu}\big)$ as follows

$$\mathbf{T}_\eta = \mathbf{P}\boldsymbol{\Psi}\mathbf{P}^T \tag{3.74}$$

and thus

$$\phi_\gamma^+(\mathbf{T}_\eta) = \mathbf{P}\text{diag}(\phi_\gamma^+(\boldsymbol{\Psi}))\mathbf{P}^T. \tag{3.75}$$

The scalar function $\phi_\gamma^+(\cdot)$ is a convex bounded function as long as the input eigenvalues are bounded. By using the above theorems, $\phi_\gamma^+(\boldsymbol{\Psi})$ is proved to be

Lipschitz continuous, and we can write the following inequality

$$\left\|\phi_{\gamma'}^{+}\left(\mathbf{T}_{\eta}^{(t)}\right)-\phi_{\gamma'}^{+}\left(\mathbf{T}_{\eta}^{(t-1)}\right)\right\|_{F}^{2} \leq l_{c}\left\|\mathbf{T}_{\eta}^{(t)}-\mathbf{T}_{\eta}^{(t-1)}\right\|_{F}^{2}, \quad \mathbf{T}_{\eta}^{(t)}=\mathbf{T}_{\eta}\left(\mathbf{L}^{(t)};\boldsymbol{\nu}_{\text{opt}}^{(t)}\right) \tag{3.76}$$

Lemma 3.4.3. *[4]: The function $\phi_{\gamma}^{+}(x)$ is Lipschitz continues with the constant $\frac{3}{2}$.*

Proof. The domain of $\phi_{\gamma}^{+}(x)$ is $\mathcal{R}$ and hence the function is everywhere differentiable, and the derivative is bounded. Therefore, the Lipschitz constant is as follows

$$\begin{aligned}\sup_{x}\left|\frac{\partial\phi_{\gamma}^{+}(x)}{\partial x}\right| &= \sup_{x}\left|\frac{x}{\sqrt{x^{2}+4\gamma}}+0.5\right| \\ &= \frac{3}{2}, \qquad\qquad x\in\mathbb{R}, \quad \gamma>0.\end{aligned} \tag{3.77}$$

□

Lemma 3.4.4. *[4]: Let $\mathbf{Z}$ be a real symmetric matrix with the eigenvalue matrix $\boldsymbol{\Psi}_{Z}$. Then, $\|\mathbf{Z}\|_{F}^{2}=\|\boldsymbol{\Psi}_{Z}\|_{F}^{2}$.*

Proof. If Ψ_{ii} is an eigenvalue of $\mathbf{Z}$, then Ψ_{ii}^{2} the eigenvalue of $\mathbf{Z}^{2}$. Therefore,

$$\begin{aligned}\|\mathbf{Z}\|_{F}^{2} &= \text{Tr}\left(\mathbf{Z}\mathbf{Z}^{T}\right) \\ &= \text{Tr}\left(\mathbf{Z}^{2}\right) \\ &= \sum_{i}\left[\boldsymbol{\Psi}_{Z^{2}}\right]_{ii} \\ &= \sum_{i}\left[\boldsymbol{\Psi}_{Z}\right]_{ii}^{2} \\ &= \|\boldsymbol{\Psi}_{Z}\|_{F}^{2}.\end{aligned} \tag{3.78}$$

□

Corollary 3.4.4.1. *[4]: The function $\phi_{\gamma}^{+}(\mathbf{Z})$ is a Lipschitz continues matrix valued function with the Lipschitz constant $l_{c}=\frac{9}{4}$.*

Proof. Considering (3.56), Lemma 3.4.3, and Lemma 3.4.4, the proof reads as follows

$$\begin{aligned}\left\|\phi_{\gamma}^{+}(\mathbf{Z}_{1})-\phi_{\gamma}^{+}(\mathbf{Z}_{2})\right\|_{F}^{2} &= \left\|\mathbf{P}\phi_{\gamma}^{+}(\boldsymbol{\Psi}_{1})\mathbf{P}^{T}-\mathbf{P}\phi_{\gamma}^{+}(\boldsymbol{\Psi}_{2})\mathbf{P}^{T}\right\|_{F}^{2} \\ &= \left\|\mathbf{P}\phi_{\gamma}^{+}\left(\boldsymbol{\Psi}_{1}-\boldsymbol{\Psi}_{2}\right)\mathbf{P}^{T}\right\|_{F}^{2} \\ &\leq \left(\frac{3}{2}\right)^{2}\left\|\mathbf{P}\boldsymbol{\Psi}_{1}\mathbf{P}^{T}-\mathbf{P}^{T}\boldsymbol{\Psi}_{2}\mathbf{P}^{T}\right\|_{F}^{2} \\ &= \frac{9}{4}\left\|\mathbf{Z}_{1}-\mathbf{Z}_{2}\right\|_{F}^{2}\end{aligned} \tag{3.79}$$

□

Theorem 3.4.5. *[4]: The proposed BTL-PPA is convergent when the following condition is met*

$$\left\|\boldsymbol{\nu}^{(t)}-\boldsymbol{\nu}^{(t_*)}\right\|_2^2 \leq \frac{1}{\eta^2 N^2}\left\|\mathbf{L}^{(t)}-\mathbf{L}^{(t_*)}\right\|_F^2, \tag{3.80}$$

where $\mathbf{L}^{(t_*)}$ *and* $\boldsymbol{\nu}^{(t_*)}$ *are respectively, the optimum primal and dual variables.*

Proof.

$$\begin{aligned}
\left\|\mathbf{L}^{(t+1)}-\mathbf{L}^{(t_*)}\right\|_F^2 &= \left\|\phi_{\gamma'}^{+}\left(\mathbf{T}_{\eta}^{(t)}\right)-\phi_{\gamma'}^{+}\left(\mathbf{T}_{\eta}^{(t_*)}\right)\right\|_F^2 \\
&\leq l_c\left\|\mathbf{T}_{\eta}^{(t)}-\mathbf{T}_{\eta}^{(t_*)}\right\|_F^2 \\
&= \frac{l_c}{(1+2\eta\sigma_e)^2}\left\|\left(\mathbf{L}^{(t)}-\mathbf{L}^{(t_*)}\right)+\eta\cdot\mathcal{B}_a\left(\boldsymbol{\nu}^{(t)}-\boldsymbol{\nu}^{(t_*)}\right)\right\|_F^2 \\
&\leq \frac{l_c}{(1+2\eta\sigma_e)^2}\Big(\left\|\mathbf{L}^{(t)}-\mathbf{L}^{(t_*)}\right\|_F^2+\eta^2N^2\left\|\boldsymbol{\nu}^{(t)}-\boldsymbol{\nu}^{(t_*)}\right\|_F^2\Big) \\
&\overset{(a)}{\leq} \frac{2l_c}{(1+2\eta\sigma_e)^2}\left\|\mathbf{L}^{(t)}-\mathbf{L}^{(t_*)}\right\|_F^2,
\end{aligned} \tag{3.81}$$

where $\mathcal{B}_a$ is the adjoint operator defined in (3.51) and $\overset{(a)}{\leq}$ holds when (3.80) is met.

According to (3.81), it is required to satisfy $\frac{2l_c}{(1+2\eta\sigma_e)^2}<1$ for the convergence which can be guaranteed by using Corollary 3.4.4.1 and having $\eta\geq\frac{3\sqrt{2}-2}{4\sigma_e}$. □

3.5 Experimental Results

Some of the most important state-of-the-art algorithms as well as our proposed algorithm are evaluated for synthetic and real data sets. For the topology learning scenarios, the performance of BTL-PPA is compared against three existing algorithms which are GL-SigRep [41], CGL [53], and learning sparse graph (LSG) [40]. These three methods have already been reviewed in Sec. 3.1. For the signal recovery performance comparison, BTL-PPA is only compared with GL-SigRep, since the other methods have no signal recovery approach. To clarify more about the state-of-the-art algorithms, here we briefly note the optimization problem that each of them is solving.

- **GL-SigRep**:

$$\begin{aligned} \underset{\mathbf{L},\mathbf{X}}{\operatorname{argmin}} \quad & \|\mathbf{Y}-\mathbf{X}\|_F^2 - \alpha \mathrm{Tr}\big(\mathbf{X}^T\mathbf{L}\mathbf{X}\big) + \beta \|\mathbf{L}\|_F^2 \\ \text{s.t.} \quad & \mathbf{L}_{ij} = \mathbf{L}_{ji}, \\ & \mathbf{L}_{ij} \leq 0 \text{ if } i \neq j, \\ & \mathbf{L}\cdot\mathbf{1}_N = \mathbf{0}_N, \\ & \mathrm{Tr}(\mathbf{L}) = c, \end{aligned} \tag{3.82}$$

  where α and β are two positive regularization parameters that have been optimized by a grid search. The algorithm to implement this minimization problem is proposed in [41].

- **CGL**:

$$\underset{\mathbf{L}}{\operatorname{argmin}} \quad -\operatorname{logdet}(\mathbf{L}+\mathbf{J}) + \mathrm{Tr}\Big(\mathbf{L}\big(\mathbf{S}+\mathbf{J}+\alpha(2\mathbf{I}-\mathbf{1}\mathbf{1}^T)\big)\Big), \tag{3.83}$$

  where α is the ℓ_1-regularization parameter to control the sparsity.

- **LSG**:

$$\begin{aligned} \underset{\tilde{\mathbf{\Delta}}\succ 0,\ \mathbf{W},\ \varsigma^2}{\operatorname{argmin}} \quad & \operatorname{logdet}\tilde{\mathbf{\Delta}} - \mathrm{Tr}(\tilde{\mathbf{\Delta}}\mathbf{S}) - \frac{\alpha}{K}\|\mathbf{W}\|_1, \\ \text{s.t.} \quad & \tilde{\mathbf{\Delta}} = \mathrm{diag}(\mathbf{W}\cdot\mathbf{1}_N) - \mathbf{W} + \frac{\mathbf{I}}{\varsigma^2}, \\ & \mathbf{W}_{ij} \geq 0, \quad i=1,\ldots,N, \quad j=1,\ldots,N \\ & \mathbf{W}_{ii} = 0, \quad i=1,\ldots,N, \\ & \varsigma^2 > 0, \end{aligned} \tag{3.84}$$

There are several performance metrics used in this thesis to compare different algorithms. One of these metrics applies for signal recovery performance comparison and the rest are used to evaluate how the graph topology learning has been successful. To check the signal recovery performance, the normalized mean square error (NMSE) is used which is defined as follows

$$\mathrm{NMSE} = \frac{1}{N\cdot K}\cdot\frac{\left\|\mathbf{Y}-\hat{\mathbf{Y}}\right\|_2^2}{\|\mathbf{Y}\|_2^2} \tag{3.85}$$

To evaluate the graph recovery performance, three metrics have been applied usually in the literature as follows [41, 93–96]

- F-measure $= \frac{2\cdot\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}$, where "precision" is the number of truly recovered edges to the number of reconstructed edges in the estimated graph, and "recall" is the number of truly recovered edges to the number of edges in the ground-truth graph. In the machine learning framework, the truly recovered edges are also called true positive (TP) and if an edge is recovered while it is not available in the ground-truth graph, it is called false positive (FP). Using this terminology, the F-measure can also be defined as follows

$$\text{F-measure} = \frac{2\text{TP}}{2\text{TP+FP+FN}} \tag{3.86}$$

  where FN denotes "False Negative". The F-measure only considers the support set of the edges, but not the weights.

- Normalized mean square deviation (NMSD) for graph edge estimation. This performance measure considers the weights of edges and is defined as follows

$$\text{NMSD} = \frac{1}{N^2} \cdot \frac{\left\|\mathbf{L} - \hat{\mathbf{L}}\right\|_2^2}{\|\mathbf{L}\|_2^2} \tag{3.87}$$

- Mutual information (MI) between the ground-truth and the estimated graph. This measure evaluates how much information about the edges has been recovered by the estimated graph. The normalized MI is defined as below

$$\text{NMI}(\mathbf{L} - \hat{\mathbf{L}}) = \frac{2\text{MI}(\mathbf{L} - \hat{\mathbf{L}})}{H(\mathbf{L}) + H(\hat{\mathbf{L}})}, \tag{3.88}$$

  where $H(\cdot)$ is the entropy of the input matrix.

3.5.1 Synthetic Data

First, the coordinates of $N = 50$ vertices are generated uniformly at random in the unit square. The edge weight between vertices i and j is extracted from the Gaussian radial basis function (RBF), i.e. $\exp\big(-d(i,j)^2/2\sigma^2\big)$ where $d(i,j)$ is the distance between the vertices. We set $\sigma = 0.5$ and threshold edges with weights smaller than 0.75. The weight matrix and then the graph Laplacian matrix are computed and normalized by its trace. Each data vector is sampled from $\mathbf{x}[k] \sim \mathcal{N}(\mathbf{0}_N, \mathbf{L}^\dagger)$ and added with an i.i.d. Gaussian noise. The measurements are $\mathbf{y}[k] = \mathbf{x}[k]+\mathbf{e}[k], \quad k = 1, \ldots, 300$, where the signal to noise ratio varies between $-5dB$ to $5dB$. These measurements are the given input to the Algorithm 3.1.

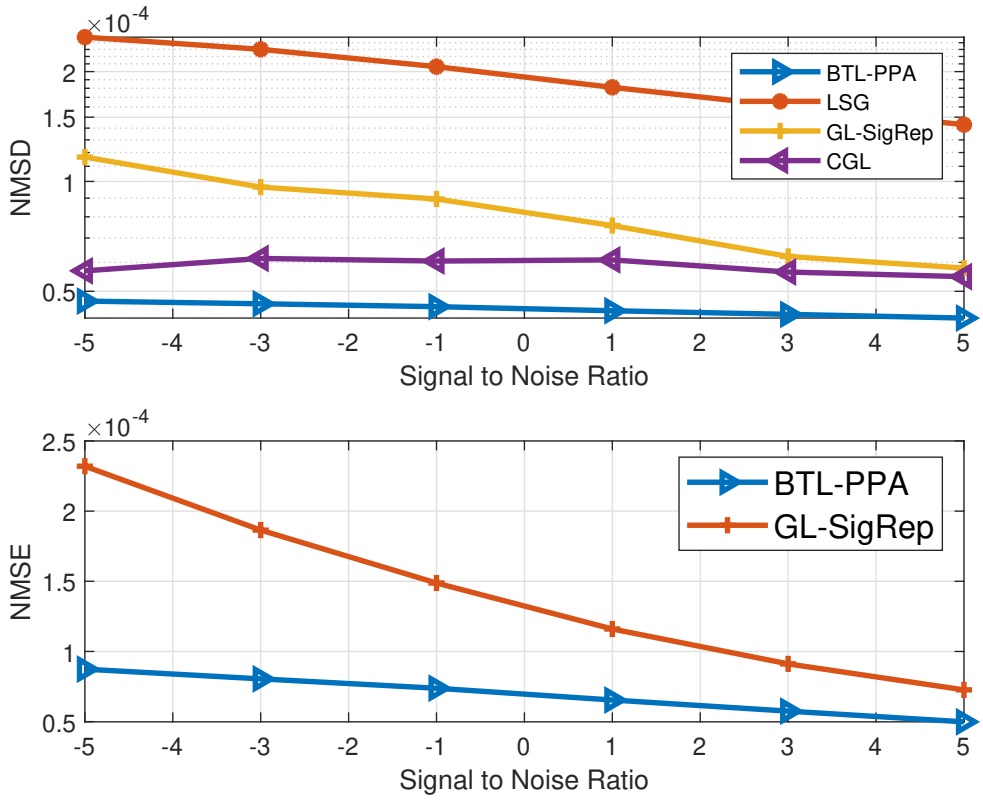


Figure 3.1: Top: The normalized mean square deviation of graph topology estimation; Bottom: The normalized mean square error of signal recovery ($N = 50$ and $K = 150$).

Figure 3.1 shows that BTL-PPA outperforms GL-SigRep for very low SNR's and its signal and topology learning errors are better than those of GL-SigRep. Considering NMSD comparison in Figure 3.1, in low SNR regimes, the BTL-PPA topology learning algorithm works very well. Still, it has a good performance for high SNR. Figure 3.2 supports the high performance of the proposed algorithm from the perspective that how many edges are captured correctly in the learned graph.

In the second simulation based on synthetic data, we run algorithms against different ratios of K/N. The SNR is kept fixed and the experiments are run for different numbers of samples, $K = \{50, 100, 150, 200, 250, 300\}$. In other words, a subset of samples is fed to the algorithms. As shown in Figure 3.3 and Figure 3.4, for large K/N ratios, the recovery algorithms provides a better performance. The intuition is that by using a larger number of samples, a better graph topology and signal recovery performance are expected. When the given samples are not sufficient, there is not enough information for learning procedures. However, the GL-SigRep's performances are reduced by utilizing more graph samples. It may be due to the noise accumulations in alternating between signal recovery and topology

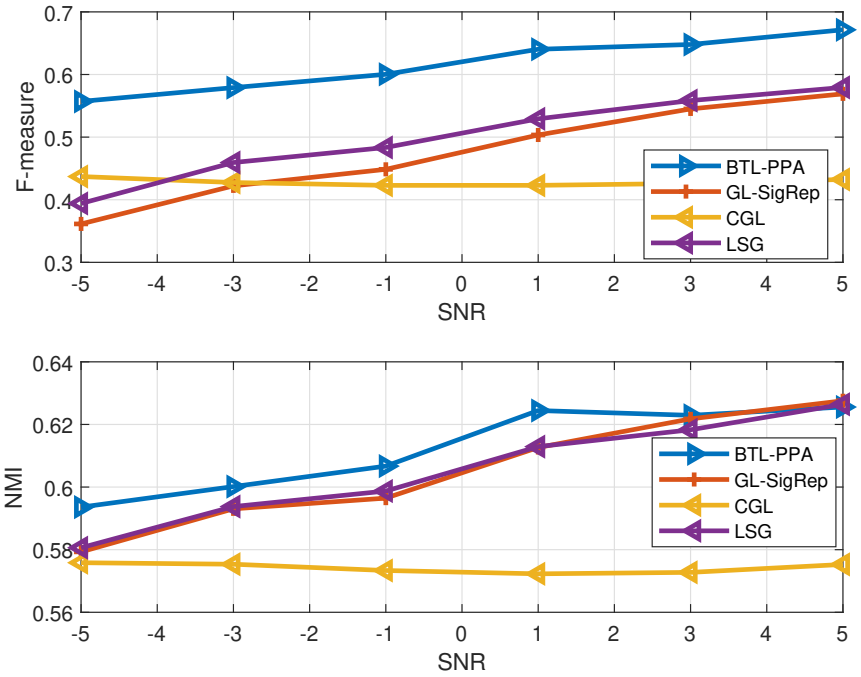


Figure 3.2: Top: F-measure; Bottom: Normalized Mutual Information (To compute the mutual information, 8-bit resolutions or 256 bins are considered to quantize the edge weights).

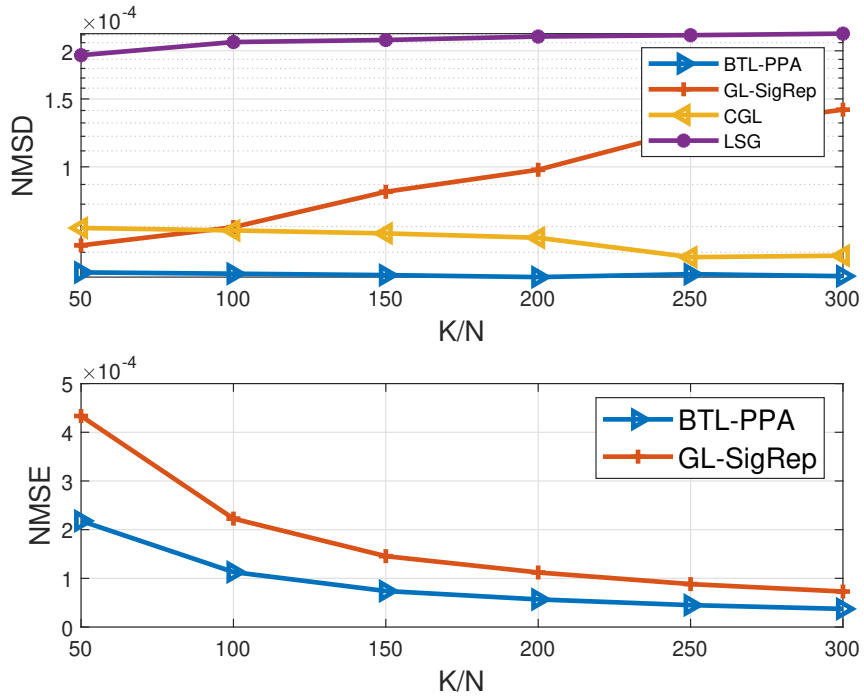


Figure 3.3: Top: The normalized mean square deviation of graph topology estimation; Bottom: The normalized mean square error of signal recovery ($N = 50$ and $SNR = -1$).

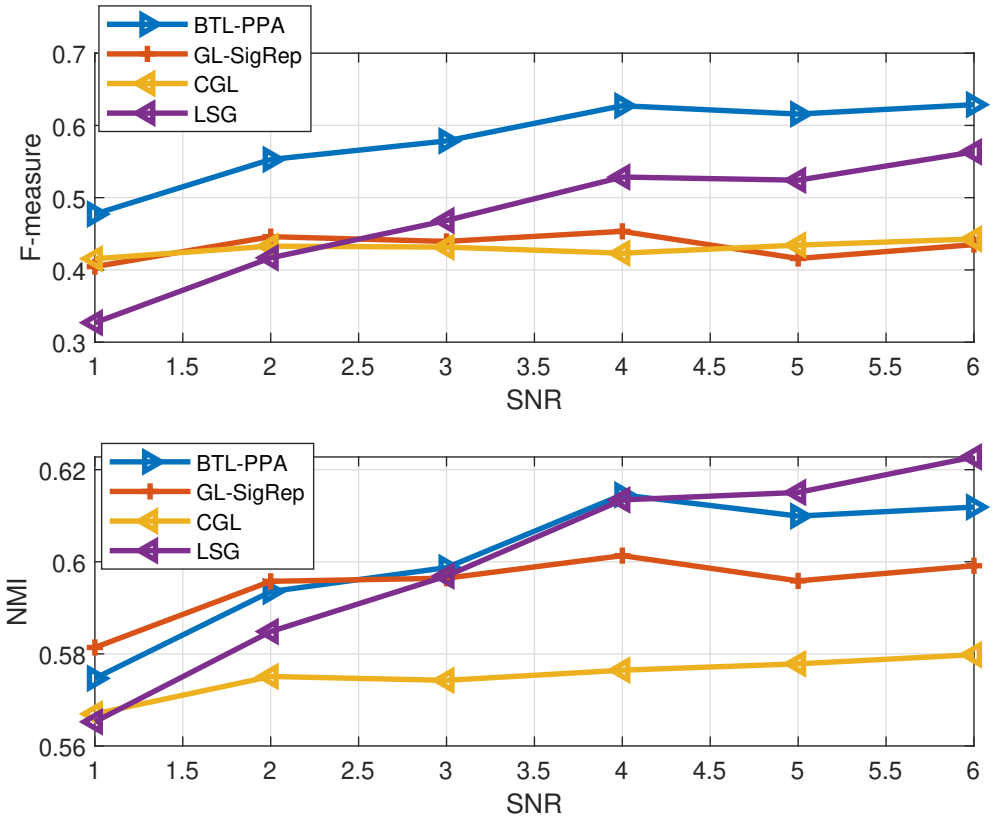


Figure 3.4: Top: F-measure; Bottom: Normalized Mutual Information (To compute the mutual information, 8-bit resolutions or 256 bins are considered to quantize the edge weights).

learning steps.

3.5.2 Temperature Data

The daily temperatures of $N = 48$ states in the USA are stored for 2011 to 2014, totally K = 1461 measurements [97]. The graph signals $\mathbf{y}_n[k]$, $k = 1, \dots 365$ are detrended by a fourth order polynomial for each $n = 1, \dots 48$ and then they are given to different learning algorithms. The graph measurements are average daily temperatures and the underlying graph topology is the propagation flow among states. The ground-truth topology is not known, but there are a couple of ways to propose it. One approach which relies on the physics of the problem is a geographically based ground-truth graph where the nodes are the states and an edge weight between two states is assumed to be the Gaussian RBF of the terrestrial distance. Following this approach to propose the ground-truth graph, Figure 3.5 compares different graph learning algorithms with respect to their ability to capture the underlying connections. The proposed BTL-PPA detects most of the edges with a good recall and

Table 3.1: Performance comparisons for different algorithms applied onto the USA temperature data. The ground-truth graph is proposed based on the physical distances.

	NMSD	NMI	F-measure
BTLPPA	10^{-4}	0.78	0.79
GL-SigRep	10^{-4}	0.72	0.69
CGL	10^{-3}	0.44	0.36
LSG	10^{-4}	0.44	0.35

Table 3.2: Performance comparisons for different algorithms applied onto the USA temperature data. The ground-truth graph is proposed based on the cross-validation.

	NMSD	NMI	F-measure
BTLPPA	10^{-8}	0.93	0.93
GL-SigRep	10^{-5}	0.90	0.88
CGL	10^{-4}	0.66	0.62
LSG	10^{-5}	0.87	0.83

precision.

Figure 3.6 shows the topology learned by our proposed method on the real map of the USA. It can be inferred that state weather flows to the neighboring states, which is also corroborated by the physics of the temperature propagation. There are more edges on the right side of the map due to the dense regions and close-by cities when compared to the left. The less number of connections in the middle of the figure can also be due to the Rocky mountains, similarly stated in [27]. The numerical results to compare different algorithms are given in Table 3.1.

In the second experiment, we used the cross-validation approach to form a ground-truth graph. Here, a portion of the dataset is used to form the ground-truth graph and the rest is used for testing purposes. The dataset is divided into two sets, training and testing data. The daily temperature of 2011 and 2012 $k = 1, \ldots, 731$[6] is used as the training data to learn the underlying topology. Then, this topology is assumed as the ground-truth graph. Now, we go for testing and use the remaining data, i.e. the data from years 2013 and 2014, is applied to estimate the topology and compare it to the first one to evaluate the consistency of the learning procedure. By consistency, it checks whether the learned topology from the training data is different from the one learned by the test data. Table 3.2 shows the results when the cross-validation is applied to propose the underlying graph.

[6]the year 2012 was a leap year

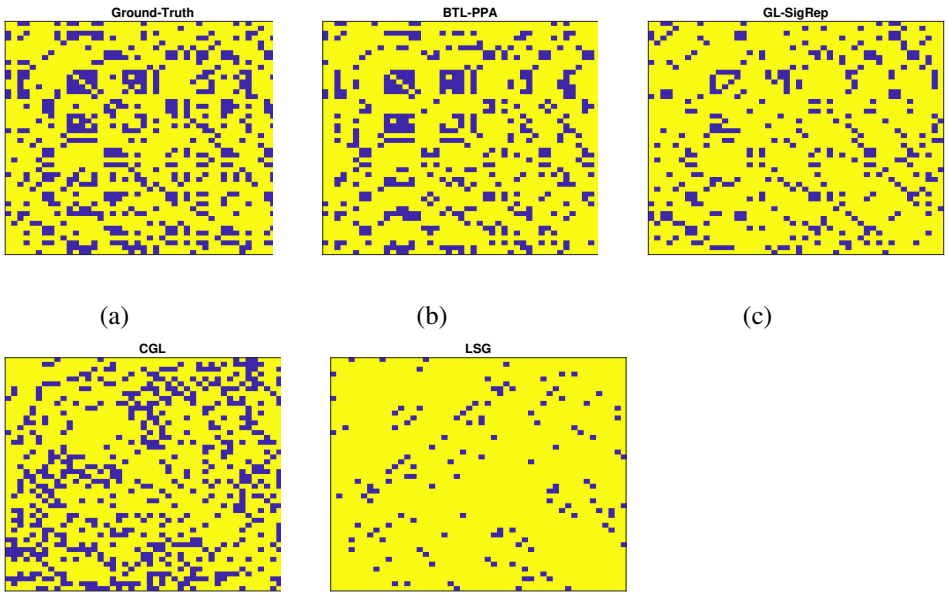


Figure 3.5: (a) the Ground-truth, and the adjacency matrix learned by (b) BTL-PPA, (c) GL-SigRep, (d) CGL, and (e) LSG. Here, blue squares show the connections between two nodes (These figures are taken from [4]).

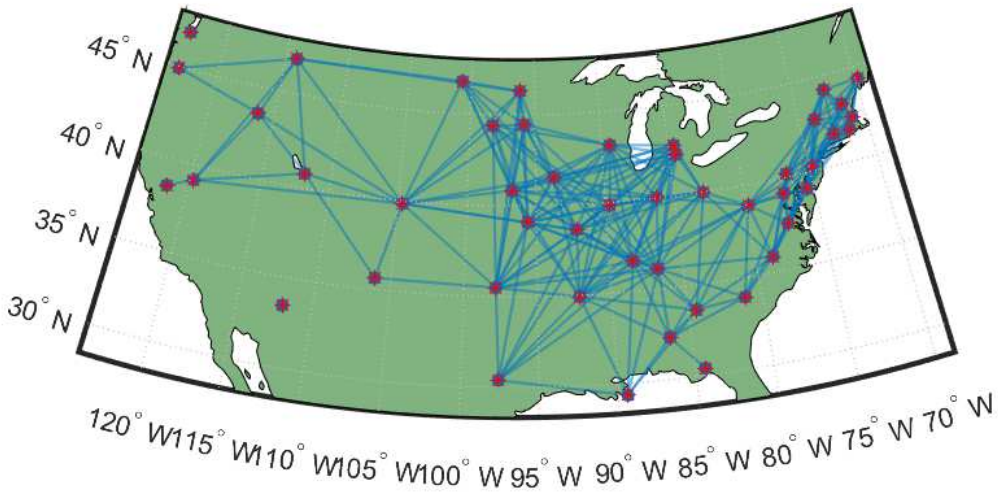


Figure 3.6: The learned topology by BTL-PPA for temperature data of 2011, mapped on the USA mainland, taken from [4].

3.6 Discussion

In this chapter, the GMRF signal model is connected to the underlying network structure and it is assumed that the observations are contaminated with an independent noise. Thus Q5 and Q6 of Sec. 1.2 have been targeted. The main optimization problem (3.39) and its fast implementation in Sec. 3.3 are respectively aimed at Q8 and Q9. Since the Gaussian signal is a widely used model in real-world scenarios, the content of this chapter can be utilized for many applications. Although some of these applications were investigated throughout the chapter, many more can be approached by the proposed algorithm and its efficient and fast implementation. The proposed online implemented algorithm is usable when we continuously receive more data over time. Moreover, the main optimization problem (3.39) is somewhat general in the sense that it can be reduced to some of the state-of-the-art algorithms.

3.7 Chapter Summary

We investigated an algorithm to explore the link between the Gaussian graph signals and the graph topology. A factor analysis model is used for signal representation in the graph domain and posterior probabilities for the signals and coefficients were computed. To formulate the problem, we used a Bayesian framework and proposed a minimum mean square estimation approach to denoise the measurements. Then, the noise variance is estimated via EM algorithm and a convex optimization problem with respect to the graph Laplacian matrix was proposed and solved via a proximal point method to estimate the topology from denoised versions of graph signals. Among the defined research questions in the first chapter, here, we approached the connection of a kind of multivariate signal processes to the graph topology (Q5), noise removal from the measurements (Q6), an optimization problem formulation for graph signal recovery (Q8), and proposing a fast algorithm to implement the solution (Q9).

Chapter 4

Robust Topology Learning for Undirected Graphs

In many applications, not only the signal is contaminated with noise, but also it is mixed with outliers. In this chapter, a method is proposed inferring the graph topology from such measurements. It is assumed that there is no information about the space graph topology, while the graph signal is sampled consecutively in the time domain and hence the graph in the time domain is given. Due to the nature of the stock market data in which noise and outliers are present, the proposed method is a good candidate to find some relations among selected ticker prices. Thus, we applied the proposed method to learn the space graph topology from some of large companies in the USA market.

One of the pioneering works in the area of undirected topology estimation is the Graphical Lasso [42], where an ℓ_1 penalty is applied to the inverse covariance matrix. The inverse covariance, so-called precision matrix, can represent the structure of the multivariate Gaussian variables. By applying a Lasso penalty to the precision matrix, it is possible to learn a network of sparse connections between some entities. However, this approach is only applicable when the underlying processes are Gaussian. Besides, the inverse covariance matrix entries can be positive or negative and do not sum to zero. In other words, this method can not satisfy the basic properties of the Laplacian matrix which can represent a graph topology. Lake and Tenenbaum [40] proposed a similar objective function applying some constraints to extract the Laplacian matrix. They applied their algorithm to learn the connection between different categories of mammals based on their features. In [41], an optimization problem is proposed to minimize the signal reconstruction error, while it

also learns a sparse graph in which the graph signals are represented smoothly on the underlying graph. The proposed method in [41] is used to estimate the connection between the altitude of a region with temperature propagation to its neighbors.

Many works are investigating the undirected topology learning from real-world data, e.g. [38, 39, 42–51, 51–53]. However, in all of them, the effect of "outliers" is not considered. In the signal processing nomenclature, the observations which significantly differ from the regular samples, are considered as outliers. There have been many works in the detection and removal of outliers, e.g. [98, 99].

In GSP, a few research projects have considered outliers estimation, but they did not learn the topology. For example, an algorithm for graph signal recovery from noisy, corrupted, and incomplete measurements is proposed in [14], but it is assumed that the graph structure is known. However, in some applications, the time-space data are highly exposed to outliers, where the underlying structure is also unknown. As an example, the stock market trades are usually dealing with insider trading, market manipulations, and other fraudulent activities which are modeled as outliers or anomalies. Insider trades refer to buy or sell shares according to some information that is not available to the public. Market manipulation means some types of oriented actions leading to decrease or increase the price for specific purposes. This is not considered as noise since "noise trading" is much about "buy" or "sell" transactions done randomly by nonprofessional traders who trade based on chances [1]. There are some research works, focusing on outliers in the stock market, e.g. [101] and [102], but they concentrated only on specific types of outliers for a given share. The price of a share may be different from time to time and hence it is possible to consider each share price by a time series of data. Moreover, the share prices affect each other in some ways. Therefore, it is possible to model the market as a network of shares and investigate the underlying topology which exists among them. The statistics also show that there are rapid and high level fluctuations modeled by outliers.

This chapter is based on the results in [6] and its main contribution is as follows; given a set of time-series measurements corrupted by noise and outliers, the main goal is to infer the underlying topology and estimate anomalies. As a real experiment, the proposed method is applied to estimate the connections among some stock market shares in the USA market.

[1] For a complete discussion about the noise in the market, the interested reader may refer to [100].

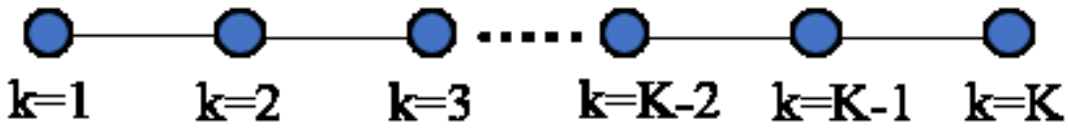


Figure 4.1: A regular time domain graph, taken from [6].

4.1 The Main Idea

It is assumed that the measurements are given as below

$$\mathbf{Y} = \mathbf{X} + \mathbf{Z} + \mathbf{E}, \tag{4.1}$$

where $\mathbf{X}$, $\mathbf{Z}$, and $\mathbf{E}$ are pure signals, noises, and outliers, respectively. The noise $\mathbf{z}[k]$ is independent of the signals and generated from an independent and identically distributed (i.i.d.) Gaussian process. We also know that $\mathbf{E}$ is sparse. The sparsity means that in each outlier vector at a specific time k, i.e. $\mathbf{e}[k]$, most of the entries are zero.

In DSP, it is assumed that samples are provided one for each time instance. Figure 4.1 shows this concept by a simple graph, connecting two consecutive time samples by an edge (the interested reader may refer to [15] for more discussions). Hereafter, we call this graph the "time graph", since it is a topology among time samples. The other graph that we used so far is called a "space graph" since it shows the topology of vertices. The time graph Laplacian is a $K \times K$ tridiagonal matrix as follows when only non-zero entries are shown [6]

$$\tilde{\mathbf{L}} = \begin{pmatrix} 1 & -1 & & & & & & \\ -1 & 2 & -1 & & & & & \\ & -1 & 2 & -1 & & & & \\ & & \ddots & \ddots & \ddots & & & \\ & & & -1 & 2 & -1 & & \\ & & & & -1 & 2 & -1 & \\ & & & & & -1 & 1 & \end{pmatrix}. \tag{4.2}$$

where for simplicity and without loss of generality, the edge weights are given as

follows

$$\tilde{\mathbf{W}}_{kk'} = \begin{cases} 1, & |k-k'| = 1 \\ 0, & |k-k'| \neq 1 \end{cases} \tag{4.3}$$

Similar to [6], the main goal of this chapter is to solve the following minimization problem

$$\begin{aligned} \underset{\mathbf{X},\mathbf{L},\mathbf{E}}{\operatorname{argmin}} \quad & \|\mathbf{Y}-\mathbf{X}-\mathbf{E}\|_F^2 + \lambda_1 \mathrm{Tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) + \lambda_2 \mathrm{Tr}\big(\mathbf{X}\tilde{\mathbf{L}}\mathbf{X}^T\big) + \lambda_3 \|\mathbf{L}\|_F^2 \\ \text{s.t.} \quad & \mathbf{L}_{ij} = \mathbf{L}_{ji}, \\ & \mathbf{L}_{ij} \leq 0 \text{ if } i \neq j, \\ & \mathbf{L}\cdot\mathbf{1}_N = \mathbf{0}_N, \\ & \mathrm{Tr}(\mathbf{L}) = c_0, \\ & \|\mathbf{E}\|_0 \leq c_1, \end{aligned} \tag{4.4}$$

where the first term encourages data fidelity. The second term controls the signal smoothness over the space while the third term promotes smoothness over time. The fourth term adjusts the off-diagonal entries of the Laplacian where $\mathrm{Tr}(\mathbf{L}) = c_0$ avoids the trivial solution for some constant c_0. The first three constraints guarantee that the learned $\mathbf{L}$ is a Laplacian matrix and the last constraint leads to a sparse outliers matrix.

Since both noise and outlier are considered in this algorithm, we call it Robust Graph Topology Learning or RGTL. It is a general algorithm for undirected graph topology inference and signal recovery due to no assumption about the process model, like Gaussianity or causality. If $\mathbf{E} = \mathbf{0}$ and $\tilde{\mathbf{L}} = \mathbf{0}$, RGTL is equivalent to GL-SigRep [41].

The minimization problem in (4.4) is convex for $\mathbf{X}$ and $\mathbf{L}$ separately and it is not convex over $\mathbf{E}$. An alternation among these three variables is used to solve (4.4). To implement the proposed algorithm, it fixes two variables in each step and solves the minimization problem for the third one. Then, it iterates over these steps up to a convergence.

Fixing $\mathbf{X}$ and $\mathbf{L}$ leads to the $\ell_2 - \ell_0$ problem as follows [6]

$$\begin{aligned} \underset{\mathbf{E}}{\operatorname{argmin}} \quad & \|\mathbf{Y}-\mathbf{X}-\mathbf{E}\|_F^2 \\ \text{s.t.} \quad & \|\mathbf{E}\|_0 \leq c_1, \end{aligned} \tag{4.5}$$

There are several methods to solve (4.5) in the sparse signal processing frame-

work. A simple one is to set $\mathbf{E} = \mathbf{Y} - \mathbf{X}$ and keep the largest entries, in absolute value sense, and set the remaining ones to zero. Thus, some columns of $\mathbf{E}$ can contain multiple non-zeros while others are just zeros. Following the procedure given in [6], by fixing $\mathbf{L}$ and $\mathbf{E}$, (4.4) reduces as follows

$$\underset{\mathbf{X}}{\operatorname{argmin}} \ \|\mathbf{Y} - \mathbf{X} - \mathbf{E}\|_F^2 + \lambda_1 \mathrm{Tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) + \lambda_2 \mathrm{Tr}\big(\mathbf{X}\tilde{\mathbf{L}}\mathbf{X}^T\big) \tag{4.6}$$

and taking its derivative and setting it to zero gives

$$\big(\mathbf{I}_N + \lambda_1\mathbf{L}\big)\mathbf{X} + \lambda_2\mathbf{X}\tilde{\mathbf{L}} = \mathbf{Y} - \mathbf{E}, \tag{4.7}$$

To find a closed form solution, the Kronecker product property helps rewriting (4.7) as the following vector form

$$\Big(\mathbf{I}_K \otimes \big(\mathbf{I}_N + \lambda_1\mathbf{L}\big)\Big)\mathrm{vec}(\mathbf{X}) + \lambda_2\big(\tilde{\mathbf{L}} \otimes \mathbf{I}_N\big)\mathrm{vec}(\mathbf{X}) = \mathrm{vec}\big(\mathbf{Y} - \mathbf{E}\big), \tag{4.8}$$

and then

$$\mathrm{vec}(\mathbf{X}) = \Big(\mathbf{I}_K \otimes \big(\mathbf{I}_N + \lambda_1\mathbf{L}\big) + \lambda_2\tilde{\mathbf{L}} \otimes \mathbf{I}_N\Big)^{-1} \cdot \mathrm{vec}\big(\mathbf{Y} - \mathbf{E}\big). \tag{4.9}$$

Then $\mathbf{X}$ and $\mathbf{E}$ are kept fixed and (4.4) is solved by the following optimization problem

$$\begin{aligned} \underset{\mathbf{L}}{\operatorname{argmin}} \ & \lambda_1 \mathrm{Tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) + \lambda_3 \|\mathbf{L}\|_F^2 \\ \text{s.t. } & \mathbf{L}_{ij} = \mathbf{L}_{ji} \\ & \mathbf{L}_{ij} \leq 0 \text{ if } i \neq j, \\ & \mathbf{L} \cdot \mathbf{1}_N = \mathbf{0}_N, \\ & \mathrm{Tr}(\mathbf{L}) = c_0. \end{aligned} \tag{4.10}$$

The Laplacian $\mathbf{L}$ is a symmetric matrix and thus this problem is solved only for entries on the lower triangular matrix. Vectorizing the lower triangular part of $\mathbf{L}$ and applying the duplication matrix $\mathcal{M}_{dup}$ [103] gives

$$\mathcal{M}_{dup} \cdot \mathrm{vech}(\mathbf{L}) = \mathrm{vec}(\mathbf{L}), \tag{4.11}$$

where $\mathrm{vech}(\mathbf{L}) \in \mathbb{R}^{\frac{N(N+1)}{2}}$ and $\mathrm{vec}(\mathbf{L}) \in \mathbb{R}^{N^2}$ denote the half-vectorization and vectorization of $\mathbf{L}$, respectively. The following identities are applicable to rewrite the objective function in the vector form as below

$$\mathrm{Tr}(\mathbf{X}^T\mathbf{L}\mathbf{X}) = \mathrm{vec}(\mathbf{X}\mathbf{X}^T)^T \cdot \mathrm{vec}(\mathbf{L}), \tag{4.12}$$

$$\|\mathbf{L}\|_F^2 = \text{vec}(\mathbf{L})^T \cdot \text{vec}(\mathbf{L}). \tag{4.13}$$

By substituting (4.11), (4.12), and (4.13) in (4.10), we have

$$\begin{aligned} \underset{\text{vech}(\mathbf{L})}{\text{argmin}}\ & \lambda_1 \text{vec}(\mathbf{X}\mathbf{X}^T)^T \cdot \mathcal{M}_{dup} \cdot \text{vech}(\mathbf{L}) + \lambda_3 \text{vech}(\mathbf{L})^T \cdot \mathcal{M}_{dup}^T \cdot \mathcal{M}_{dup} \cdot \text{vech}(\mathbf{L}) \\ \text{s.t. } & \mathbf{B} \cdot \text{vech}(\mathbf{L}) \leq \mathbf{0}_{\frac{N(N-1)}{2}}, \\ & \left(\mathbf{1}_N^T \otimes \mathbf{I}_N\right) \cdot \mathcal{M}_{dup} \cdot \text{vech}(\mathbf{L}) = \mathbf{0}_N \\ & \left(\text{vec}(\mathbf{I}_N)\right)^T \cdot \mathcal{M}_{dup} \cdot \text{vech}(\mathbf{L}) = c_0. \end{aligned} \tag{4.14}$$

where $\mathbf{B}$ manages the inequality constraint in (4.10). The problem (4.14) is a quadratic convex problem and is solved efficiently by some off-the-shelf methods [91, 104]. Here, we used CVX, a package for specifying and solving convex programs [105].

4.2 Experimental Results

The RGTL algorithm is tested numerically and compared with the state-of-the-art algorithm in [41].

4.2.1 Synthetic Data

An Erdős Rényi graph is generated with different number of vertices $N \in \{20, 25, 30, 35, 40\}$ and an edge probability of 0.2. The graph Laplacian is normalized by its trace. A set of graph signals based on multivariate Gaussian processes is generated where its mean is a zero vector and its covariance is the pseudo inverse of the graph Laplacian. Each $\mathbf{x}[k], k = 1, \ldots, 1000$ is a graph signal where the data matrix $\mathbf{X}$ stacks them in columns. The signal to noise ratio for measurements is 10dB. The outliers matrix $\mathbf{E}$ is sampled from uniformly distributed random entries with around $p_{\text{nonZero}} * N * K$ nonzero elements, where the probability of nonzero element is one percent. The regularization parameters $\lambda_1, \lambda_2, \lambda_3$ and constants c_0, c_1 are selected by grid searches over different sets of values. The simulation is run for 100 trials and the results are averaged.

Figure 4.2 shows that RGTL outperforms in several graph orders. A higher graph order leads to a lower NMSD. When the dimension of graph signals is higher, the outlier vectors are estimated easier and with more accuracy. Table 4.1 shows performance comparisons by averaging the results over different graph orders. The RGTL algorithm has a better performance numerically when compared to GL-

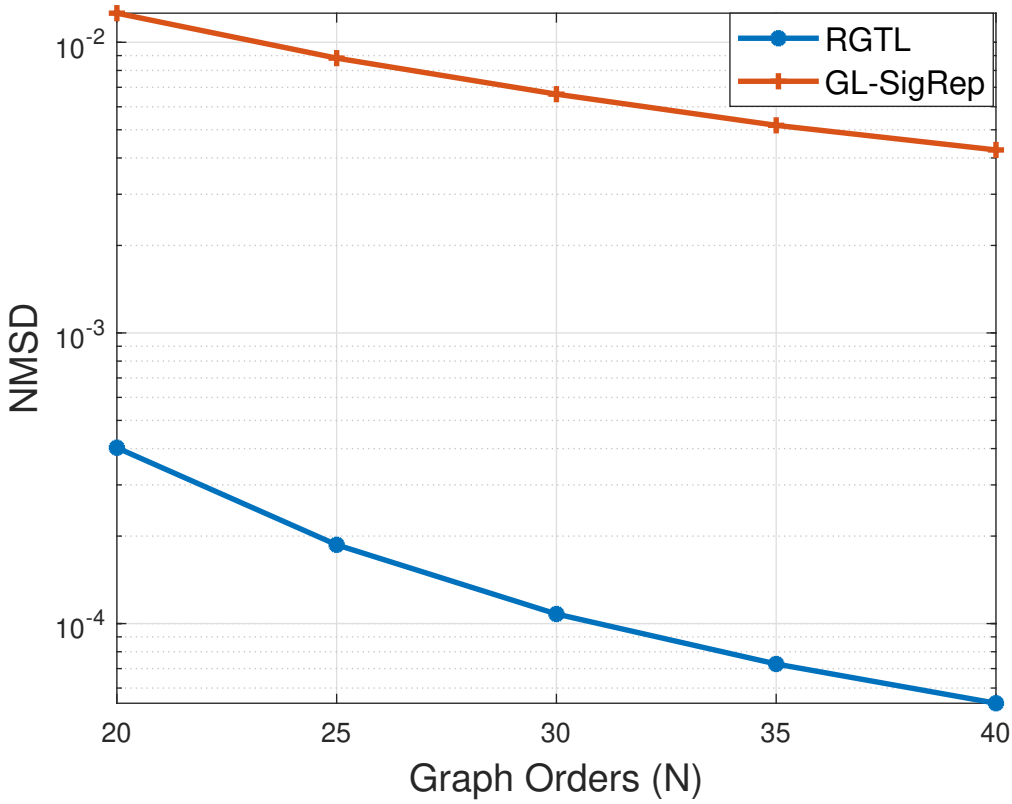


Figure 4.2: NMSD comparisons for several network orders ($K = 1000$), taken from [6].

SigRep, even for the explained scenario in which the synthetic data generation setup is what was proposed in [41] designed for GL-SigRep.

Table 4.1: The performance measures for different numbers of vertices ($K = 1000$ and $SNR = 10dB$). adopted from [6].

	NMSE	Precision	Recall	F-measure
RGTL	0.0016	0.75	0.78	0.76
GL-SigRep	0.0075	0.21	0.22	0.22

4.2.2 Real Stock Market Data

The 30 largest companies from the USA market are selected and their prices from November 2013 for $K = 1250$ days[2] are taken from NASDAQ database[3]. The market price data includes four different values as *open*, *close*, *low*, and *high*. Here,

[2]for simplicity and without loss of generality, it is assumed that each year has 250 working days and hence the entire data set is for 5 years

[3]https://www.nasdaq.com/quotes/historical-quotes.aspx

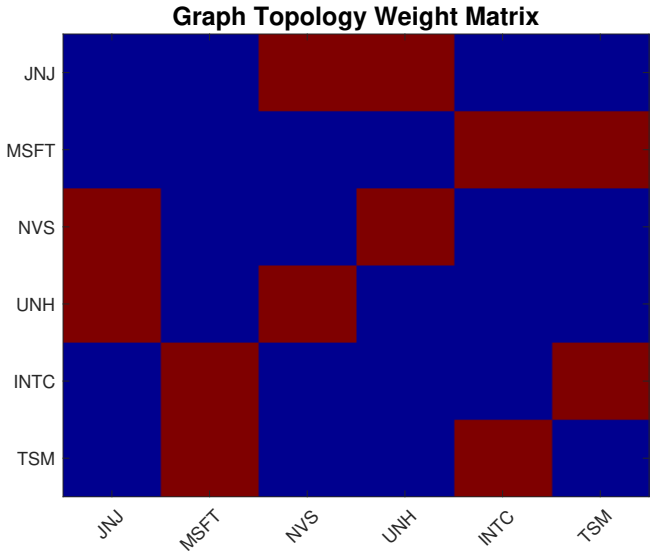


Figure 4.3: The Laplacian matrix of market data, learned by RGTL and given in [6].
The electronic based tickers: MSFT (Microsoft Corporation), INTC (Intel Corporation), and TSM (Taiwan Semiconductor Manufacturing Company),
The health-based tickers: JNJ (Johnson & Johnson), NVS (Novartis), and UNH (United-Health Group).

it is only focused on the highest price of each share per day to include the highest fluctuations.

After normalizing by ℓ_2-norm, we evaluate how the learned topology from a train data set is consistent with the one from the test data set, applying cross-validation. The first half of the data is served for the graph learning stage and the remaining data is utilized for the test. In other words, first, a graph is learned from the shares and then we evaluate whether this graph can be applied for a future prediction. The performance measures values are shown in Table 4.2. Due to the robustness of RGTL with respect to the market's shocking fluctuations, it provides a better NMSD and F-measure. The results support RGTL consistency for learning the structure of the network.

Table 4.2: Performance comparisons for the stock market shares data, adopted from [6].

	NMSD	F-measure
RGTL	3.57×10^{-6}	0.80
GL-SigRep	6.63×10^{-4}	0.59

In the next simulation, three companies from the electronic industry and three ones from the health-related category are selected. Figure 4.3 shows how these six shares are linked to each other. It is shown that *Microsoft* is related to *Intel* and

Taiwan Semiconductor, but not the other ones. This is probably because of the different nature of "Electronic" and "health" categories in the reality, reflected also in the market shares.

4.3 Discussion

In the same way as Chapter 3, this chapter also answers research questions Q5, Q6, and Q8 of Sec. 1.2. Following the main question of the thesis, the graph topology learning is investigated for a general signal model which targets Q5. Not only the observation has been contaminated by noise (Q6), but also the main focus of this chapter was on the outliers, discussed in Q7. By solving the main optimization problem of (4.4), we estimated the graph signals, the space graph Laplacian matrix, and outliers when we have a priori knowledge about the time order graph.

4.4 Chapter Summary

We investigated an approach to learn the underlying topology when the given data set is contaminated with noise and outliers. One of the applications for the proposed algorithm is the market shares dependencies, where each share is modeled as an entity in the information network and its daily prices as a time-series over that entity. Because of the sparsity of the outliers, an ℓ_0-norm constraint is added to the optimization problem, where data fidelity and signal smoothness are also considered. The proposed algorithm iteratively solves the problem as given in eq. (4.4). It learns the Lapacian matrix $\mathbf{L}$ using eq. (4.14), estimates the outliers $\mathbf{E}$ using eq. (4.5), and recovers the signal $\mathbf{X}$ using eq. (4.9). Regarding the defined research questions in the first chapter, we approached the connection of a specific type of multivariate signal processes to the underlying topology (Q5), noise contamination in the measurements (Q6), outlier removal from graph measurements (Q7), and an optimization problem formulation for graph prediction (Q8).

Chapter 5

Undirected Topology Inference via Dictionary Learning

In this chapter, we investigate dictionary learning concepts in the graph signal processing framework. A Joint Graph Learning and Signal Representation algorithm is proposed for simultaneous topology learning and graph signal representation. This approach is done via a learned over-complete dictionary on the underlying graph structure. The proposed algorithm alternates among three main phases: sparse coding, dictionary learning, and graph topology learning. We also introduce the "transformed graph", a projected graph into the transform space consisting of dictionary atoms as bases. The experimental results with the synthetic data set confirm the superiority of the proposed method to sparsely represent the graph signals by a dictionary. Besides, by testing the proposed method on real temperature data set, we realize that it is a good candidate to model weather propagation flow from one region to its neighbors.

Many problems in signal processing and machine learning are dealing with the high dimensional data, i.e. $\mathbf{y} \in \mathbb{R}^N$ for large N. A bunch of K signals is represented as an $N \times K$ data matrix $\mathbf{Y}$ and then the matrix factorization gives a significant contribution in the data analysis such as data compression, dimensionality reduction, and information retrieval [106–109]. The above methods can not capture the possible nonlinearity or structure underlined the input data. As we discussed in the previous chapters, for such scenarios, a standard way to represent pairwise connections between entities is by using affinity graphs.

In [110], an affinity graph is applied to encode the geometrical structure of the

data and a graph regularized non-negative matrix factorization algorithm is proposed. Shahid and his colleagues utilized the Laplacian matrix and applied the principal component analysis (PCA) approach to recover the low-rank data representation [111]. In [112], an approach is investigated to learn the sparse representation of the data when the local structure is considered. There have been many more research works following geometrical structure for the data representation, e.g. [113–118]. However, these approaches assume that the underlying topology is known. In [54], a graph Dictionary Learning approach is proposed for graph topology inference, dictionary learning, and sparse coding at the same time. In [66], a general framework for graph topology inference via transform learning is proposed, satisfying the dictionary completeness and the orthonormality of atoms criteria.

The main contribution of this chapter is to propose a new algorithm for Joint Graph Learning and Signal Representation (JGLSR) which has the following capabilities (This chapter is based on the results in [120]):

- Estimating the set of dictionary atoms,
- Finding the sparse coefficients which represent the signals in the transform domain,
- Learning the graph topology of the data space, revealing the underlying geometry of the data domain.

The important features and main assumptions of JGLSR are as follows

- There is no assumption of the a priori knowledge for the dictionary,
- Contrary to [41], our proposed approach does not assume that the data is Gaussian,
- JGLSR minimizes the average of atom coherence, while the existing method in [54] uses some empirical regularization parameters to have a dictionary with reasonable coherence,
- In JGLSR, the dictionary is not the eigenvector matrix of the Laplacian matrix. Thus, the graph signals may not share a common support. In this respect, it is different from the transform learning method proposed in [66].

5.1 Dictionary Learning Introduction

In simple words, dictionary learning is the process of learning a matrix called ”dictionary” in a way that we can represent a signal by a linear combination of the

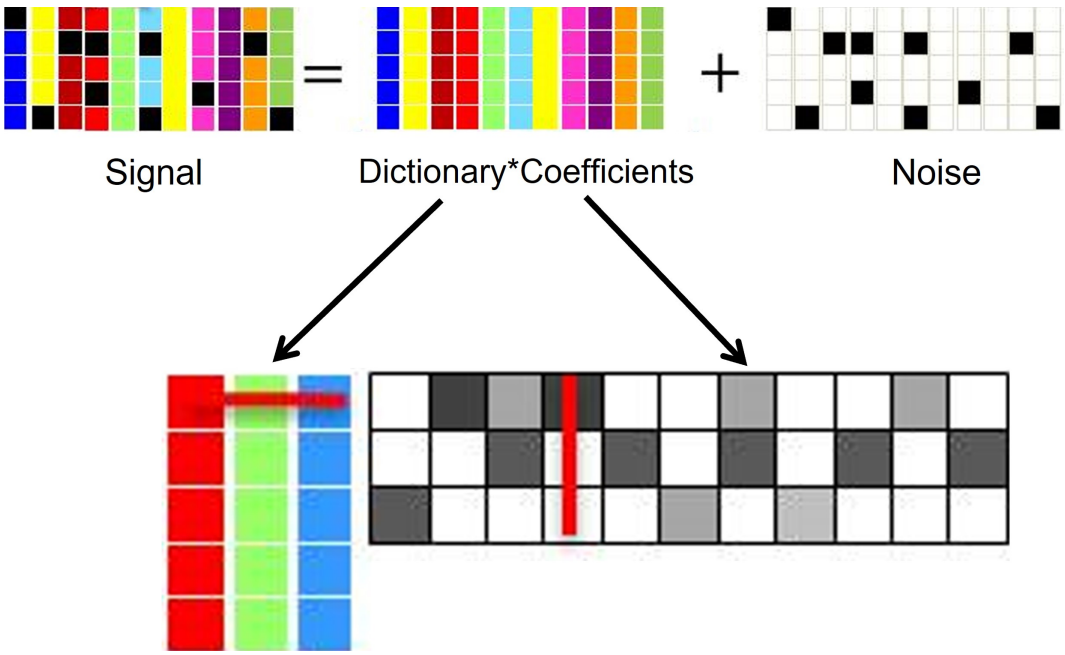


Figure 5.1: A simple illustration for the computational framework of dictionary learning and sparse coding.

matrix columns, called atoms. The aim of this framework is to find a sparse representation of the signal of interest. In other words, we try to find a dictionary in a way that the given signal can be represented by just a few of its atoms. Hence, the process of finding the sparse coefficient vector in the dictionary domain is called "sparse coding" (see Figure 5.1).

For example, any image can be shown by a linear combination of only a few basic features and if these features are stored in the columns of a matrix, we can sparsely represent the image of interest. If the number of atoms is greater than the dimension of each atom, the dictionary is called over-complete (and vice versa). An over-complete dictionary permits a signal to be sparsely represented by its atoms. To illustrate this process by a practical application, the example of whole-brain rsfMRI signals for identification of functional brain networks [7] is given in Figure 5.2.

To find the dictionary and coefficients, an optimization problem is usually formulated in which the objective function includes the data fidelity term, the sparsity condition over the coefficient vector, and some constraints on the set of atoms. As usual, the data fidelity term concerns the error between the original signal and its representation in the dictionary domain. Also, the constraints on atoms should satisfy their incoherence with each other, i.e. each atom can reflect a portion of signal information when the others can not. In other words, the amount of common infor-

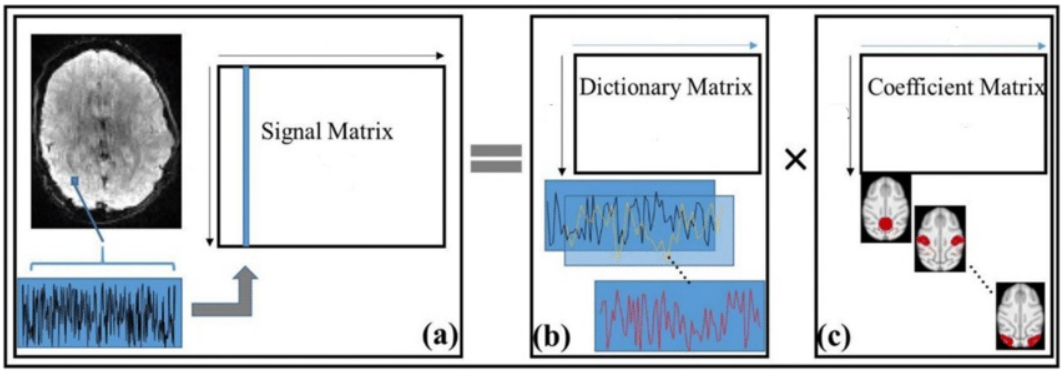


Figure 5.2: Dictionary learning application in functional brain network identification; (a) The rsfMRI data matrix for one subject, (b) The computed dictionary matrix whose each column is a temporal pattern, and (c) The sparse codes whose each entity denotes the functional activity value (for more details on this example, please refer to [7]).

mation that two atoms represented for a signal must be as low as possible.

5.2 The Main Idea

The data matrix $\mathbf{Y}$ is given as below:

$$\mathbf{Y} = \mathbf{U}\mathbf{X} + \mathbf{E}, \tag{5.1}$$

where $\mathbf{X} \in \mathbb{R}^{M_D \times K}$ and $\mathbf{E}$ are the sparse coefficient and error matrices, respectively, and $N < M_D < K$ which makes $\mathbf{U} \in \mathbb{R}^{N \times M_D}$ to be an over-complete dictionary. The proposed algorithm, adopted from [120], solves the following optimization problem

$$\begin{aligned}
\underset{\mathbf{U},\mathbf{X},\mathbf{L}}{\operatorname{argmin}} \; & \|\mathbf{Y} - \mathbf{U}\mathbf{X}\|_F^2 + \alpha_1 \mathrm{Tr}\big((\mathbf{U}\mathbf{X})^T \mathbf{L} \mathbf{U}\mathbf{X}\big) + \alpha_2 \sum_{k=1}^{K} \|\mathbf{x}[k]\|_1 + \alpha_3 \|\mathbf{L}\|_F^2 \\
& + \alpha_4 r(\mathbf{U}) \\
\text{s.t.} \; & \mathbf{L}_{ij} = \mathbf{L}_{ji}, \\
& \mathbf{L}_{ij} \leq 0 \text{ if } i \neq j, \\
& \mathbf{L} \cdot \mathbf{1}_N = \mathbf{0}_N, \\
& \mathrm{Tr}(\mathbf{L}) = c_0, \\
& \mathrm{diag}(\mathbf{U}^T \mathbf{U}) = \mathbf{1}_M,
\end{aligned} \tag{5.2}$$

where $\alpha_i \in \mathbb{R}^+$, $i = 1, ..., 4$ are regularization parameters. The first term of the objective function promotes the data fidelity and the second term encourages smoothness of the represented signals in the dictionary domain on the underlying graph. The third term promotes coefficient sparsity. When the constraint $\text{Tr}(\mathbf{L}) = c_0$ for some constant c_0 controls the diagonal elements and avoids the trivial solution, the fourth term of the objective function controls the off-diagonal entries of the Laplacian matrix. The first three constraints guarantee the validity of learned $\mathbf{L}$ as a graph Laplacian. Moreover, these constraints prevent the identifiability issue. The last constraint normalizes the dictionary atoms and the last term of the objective function encourages low coherence of atoms, given as below [119, 120]

$$r(\mathbf{U}) := - \sum_{1 \leq m < m' \leq M_D} \log(1 - (\mathbf{u}[m]^T \mathbf{u}[m'])^2). \tag{5.3}$$

In the proposed approach, to have a better performance, the importance of dictionary coherence and represented signal smoothness (instead of observed signal smoothness) are considered. The objective function of (5.2) is convex for each variable separately. In other words, if any two of $\mathbf{U}$, $\mathbf{L}$, and $\mathbf{X}$ are kept fixed, the objective function in (5.2) is convex with respect to the third one. Thus, a local minimum can be achieved via an alternating method.

In 5.2.1, $\mathbf{L}$ and $\mathbf{U}$ are fixed and the sparse coding problem is solved. Then, $\mathbf{L}$ and $\mathbf{X}$ are kept fixed in 5.2.2 and the dictionary is estimated. Finally, (5.2) is solved with respect to $\mathbf{L}$ in 5.2.3 to learn the graph topology.

5.2.1 Sparse Coefficients Estimation

The goal is to solve the following problem, taken from [120]

$$\underset{\mathbf{X}}{\text{argmin}} \, \|\mathbf{Y} - \mathbf{U}\mathbf{X}\|_F^2 + \alpha_1 \text{Tr}(\mathbf{X}^T \tilde{\mathbf{L}} \mathbf{X}) + \alpha_2 \sum_{k=1}^{K} \|\mathbf{x}[k]\|_1 . \tag{5.4}$$

where $\tilde{\mathbf{L}} = \mathbf{U}^T \mathbf{L} \mathbf{U}$ is the transformed graph Laplacian into the dictionary space. The signal smoothness in the measurement domain is considered as the coefficients smoothness in the dictionary domain. Following (2.16) gives

$$\tilde{\mathbf{L}} = \mathbf{U}^T \boldsymbol{\chi} \boldsymbol{\Lambda} \boldsymbol{\chi}^T \mathbf{U} = \widehat{\mathbf{U}} \boldsymbol{\Lambda} \widehat{\mathbf{U}}^T, \tag{5.5}$$

where $\widehat{\mathbf{U}} = \mathbf{U}^T \boldsymbol{\chi}$ is the GFT of dictionary atoms. For an orthonormal complete dictionary $\mathbf{U} \in \mathbb{R}^{N \times N}$, the eigenvalues of the Laplacian and the transformed matri-

ces are identical and the eigenvectors of the transformed Laplacian are the GFTs of the dictionary atoms. It is not possible to apply this simplification here as long as the dictionary is over-complete, unlike in [66].

The objective function of (5.4) is convex but it is not differentiable due to the ℓ_1-norm. Hence, a standard unconstrained minimization approach has not been used to find a closed form solution. Following the algorithm in [120], the Alternating Direction Method of Multipliers (ADMM) [104] is applied to propose a solution. Thus, the problem is rewritten as follows,

$$\underset{\mathbf{x}[1],\ldots,\mathbf{x}[K]}{\operatorname{argmin}} \sum_{k=1}^{K} \|\mathbf{y}[k] - \mathbf{U}\mathbf{x}[k]\|_2^2 + \alpha_1 \sum_{k=1}^{K} \mathbf{x}^T[k]\tilde{L}\mathbf{x}[k] + \alpha_2 \sum_{k=1}^{K} \|\mathbf{x}[k]\|_1 . \tag{5.6}$$

where the objective is separable over $\mathbf{x}[k]$'s. To solve (5.6) for the k'th variable, we have the following vector optimization problem,

$$\underset{\mathbf{x}[k]}{\operatorname{argmin}} \|\mathbf{y}[k] - \mathbf{U}\mathbf{x}[k]\|_2^2 + \alpha_1 \mathbf{x}^T[k]\tilde{L}\mathbf{x}[k] + \alpha_2 \|\mathbf{x}[k]\|_1 . \tag{5.7}$$

In the ADMM algorithm, the non-differentiable term is removed and (5.7) is rewritten as follows

$$\begin{aligned} &\underset{\mathbf{x}[k],\mathbf{z}[k]}{\operatorname{argmin}}\ l(\mathbf{x}[k]) + g(\mathbf{z}[k]) \\ &\text{s.t. } \mathbf{x}[k] - \mathbf{z}[k] = \mathbf{0}_{M_D}, \end{aligned} \tag{5.8}$$

where

$$l(\mathbf{x}[k]) = \|\mathbf{y}[k] - \mathbf{U}\mathbf{x}[k]\|_2^2 + \alpha_1 \mathbf{x}^T[k]\tilde{L}\mathbf{x}[k], \tag{5.9}$$

and

$$g(\mathbf{z}[k]) = \alpha_2 \|\mathbf{z}[k]\|_1 . \tag{5.10}$$

Adopted from [120], the scaled form of the ADMM algorithm follows these steps:

$$\mathbf{x}^{\tau+1}[k] := \underset{\mathbf{x}[k]}{\operatorname{argmin}} \Big(l(\mathbf{x}[k]) + \frac{\rho}{2} \|\mathbf{x}[k] - \mathbf{z}^{\tau}[k] + \mathbf{v}^{\tau}[k]\|_2^2 \Big), \tag{5.11a}$$

$$\mathbf{z}^{\tau+1}[k] := \underset{\mathbf{z}[k]}{\operatorname{argmin}} \Big(g(\mathbf{z}[k]) + \frac{\rho}{2} \big\|\mathbf{z}[k] - \mathbf{x}^{\tau+1}[k] - \mathbf{v}^{\tau}[k]\big\|_2^2 \Big), \tag{5.11b}$$

$$\mathbf{v}^{\tau+1}[k] := \mathbf{x}^{\tau+1}[k] + \mathbf{v}^{\tau}[k] - \mathbf{z}^{\tau+1}[k], \tag{5.11c}$$

where $\rho > 0$, τ and $\mathbf{v}[k]$ denote the penalty parameter, the ADMM iteration index, and the scaled dual variable, respectively [104]. To solve (5.11a), the following

system of equation is obtained by a simple derivation,

$$\Big(\mathbf{U}^T\mathbf{U} + \frac{\rho}{2}\mathbf{I}_M + \alpha_1\tilde{\mathbf{L}}\Big)\mathbf{x}[k] = \mathbf{U}^T\mathbf{y}[k] + \frac{\rho}{2}(\mathbf{z}^\tau[k] - \mathbf{v}^\tau[k]). \tag{5.12}$$

Due to the positive definiteness of the left-hand side matrix, the Cholesky factorization solves it efficiently. For (5.11b), a closed form solution is achieved by applying subdifferential calculus [121] as below

$$\mathbf{z}^{\tau+1}[k] = \mathcal{S}_{\frac{\alpha_2}{\rho}}(\mathbf{x}^{\tau+1}[k] + \mathbf{v}^\tau[k]), \tag{5.13}$$

where the element-wise soft thresholding $\mathcal{S}_\kappa(a)$ is the proximal operator of the ℓ_1-norm which is given as follows

$$\mathcal{S}_\kappa(a) = \begin{cases} a-\kappa, & a > \kappa \\ 0, & |a| \le \kappa \\ a+\kappa & a < -\kappa. \end{cases} \tag{5.14}$$

5.2.2 Dictionary Learning

To find $\mathbf{U}$, [120] proposed to rewrite (5.2) as follows

$$\begin{aligned} \underset{\mathbf{U}}{\text{argmin}}\ & \|\mathbf{Y} - \mathbf{U}\mathbf{X}\|_F^2 + \alpha_1 \text{Tr}(\mathbf{X}^T\mathbf{U}^T\mathbf{L}\mathbf{U}\mathbf{X}) + \alpha_4 r(\mathbf{U}), \\ & \text{s.t. } \text{diag}(\mathbf{U}^T\mathbf{U}) = \mathbf{1}_{M_D}. \end{aligned} \tag{5.15}$$

where the constraint forces atoms to be normalized. This minimization problem can be solved by usual optimization toolboxes. In our implementation and simulations, it is solved by the L-BFGS algorithm [122]. First, we can find the minimizer of the objective function using L-BFGS and then normalize each column to have unit ℓ_2-norm.

5.2.3 Graph Topology Learning

With respect to $\mathbf{L}$, we have the following optimization problem [120]

$$\begin{aligned} \underset{\mathbf{L}}{\text{argmin}} \quad & \alpha_1 \text{Tr}(\mathbf{X}^T\mathbf{U}^T\mathbf{L}\mathbf{U}\mathbf{X}) + \alpha_3 \left\|\mathbf{L}\right\|_F^2 \\ \text{s.t.} \quad & \mathbf{L}_{ij} = \mathbf{L}_{ji}, \\ & \mathbf{L}_{ij} \leq 0 \text{ if } i \neq j, \\ & \mathbf{L} \cdot \mathbf{1}_N = \mathbf{0}_N, \\ & \text{Tr}(\mathbf{L}) = c_0. \end{aligned} \tag{5.16}$$

This minimization problem is similar to (4.10) and can be solved in the same way presented in 4.1.

5.3 Experimental Results

The proposed method is tested on a synthetic data set and also a real temperature data set from the USA mainland [97].

5.3.1 Synthetic Data

The data is constructed using a Gaussian RBF following the approach discussed in [54]. The coordinates of $N = 25$ vertices are generated uniformly at random in the square $[0, \sqrt{5}] \times [0, \sqrt{5}]$ and the edge weights are determined by $\exp\left(-d(i,j)^2/2\sigma^2\right)$ where $d(i,j)$ is the distance of vertices i and j. The width of the RBF is set $\sigma = 0.5$ and the edges with weights less than 0.5 are removed to keep around 17% of edges. Thus, we have a fairly sparse graph. The graph Laplacian is normalized by its trace. To have a smooth dictionary on this topology, an initial random dictionary $\mathbf{U}_0$ is generated and then it is filtered by a first order LSI as follows

$$\mathbf{U} = \left(\mathbf{I}_N + \lambda\mathbf{L}\right)^{-1}\mathbf{U}_0, \tag{5.17}$$

where $\lambda = 5$ [54]. A random coefficient matrix $\mathbf{X} \in \mathbf{R}^{N\times K}$ for $K = 1000$ is drawn and its product by the dictionary is contaminated with an independent noise. By changing SNR, we have $\mathbf{Y}$ of (5.1). Given $\mathbf{Y}$, the goal is to learn the underlying topology via JGLSR, graphDL [54], and GL-SigRep [41]. The results are compared by using the five performance measures presented in 4.2. The results are averaged over 100 trials. Figure 5.3 shows the higher performance of JGLSR for different

Table 5.1: Performance comparisons for synthetic data simulation, taken from [120].

	Recall	Precision	F-measure
JGLSR	0.90	0.87	0.89
graphDL	0.61	0.32	0.42
GL-SigRep	0.36	0.99	0.52

SNR. Table 5.1 also compares three performance measures by averaging the results over different SNRs.

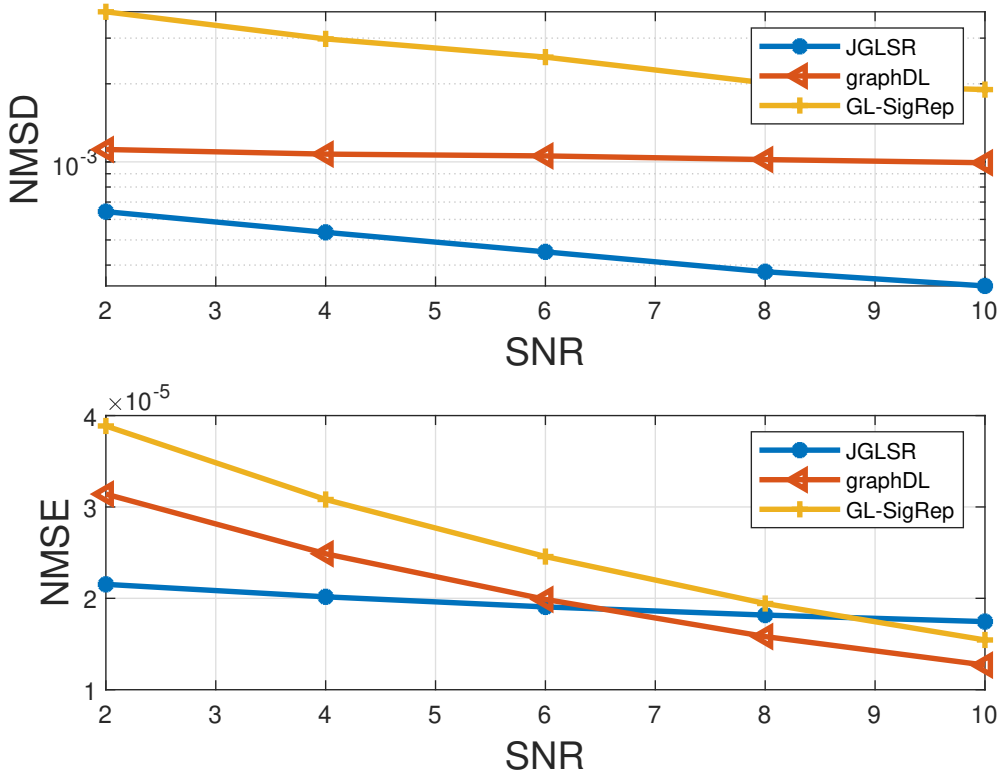


Figure 5.3: NMSD and NMSE for synthetic data. The number of vertices is 25 and a Gaussian RBF is used to weigh the edges. The figures are adopted from [120].

5.3.2 Temperature Data

The average temperatures of $N = 48$ states for 2011, 2012, and 2013 are stored daily, hence $K = 1096$ samples. Contrary to the first simulation, there is no access to the ground-truth graph, but we can consider a geographical-based graph as a true one. In this respect, a graph whose node set includes states is constructed. Also, the edge weights are computed by the Gaussian RBF of the distance between every

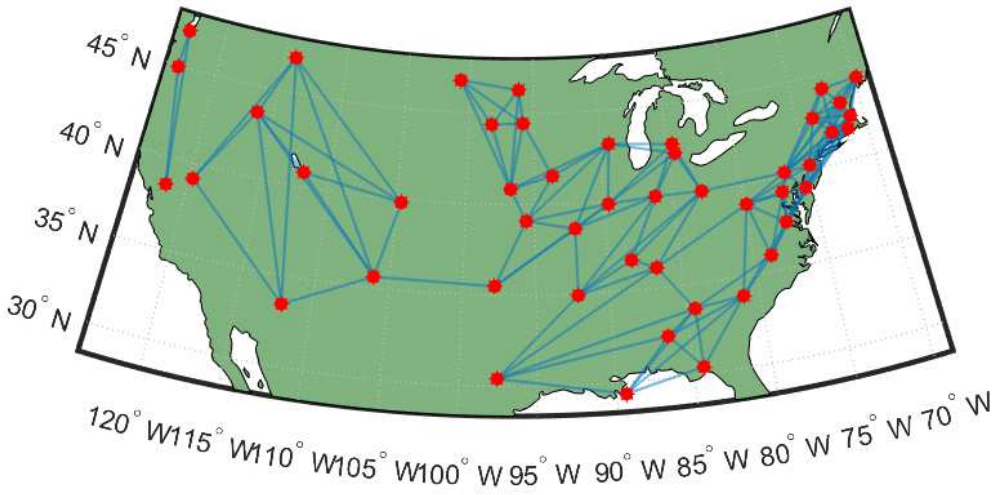


Figure 5.4: The learned Laplacian of real temperature data depicted on the USA map, taken from [120].

two states. The parameters for the proposed algorithm are set to $\alpha_1 = 0.001, \alpha_2 = 0.1, \alpha_3 = 5, \alpha_4 = 100$, and $\sigma = 0.001$. The F-measure for GL-SigRep, graphDL, and JGLSR are 0.51, 0.59, and 0.64, respectively. The NMSD of all methods is in the order of 10^{-4}.

Figure 5.4 shows the learned graph, projected on the map of the USA mainland. Considering the learned graph, it is inferred that a state's weather affects the neighboring states, which is also intuitive from the physical point of view. In other words, the closer the two states are, the stronger connection exists between them. The concentration of edges is on the right side of the map, explaining that there are more cities in the neighborhood, which is contrary to the west coast. The few edges in the middle of the map are probably because of the Rocky mountain which prevents an easy temperature flow.

5.4 Discussion

In this chapter, we use the product of the dictionary and coefficient matrices to model the graph signals. By connecting the concept of learning a dictionary to the

graph topology inference, Q3 of Sec. 1.2 is targeted. Besides, by elaborating on the atom coherence and using the last term of the optimization problem (5.2), the fourth question of Sec. 1.2, i.e. Q4 is answered. Similar to the previous chapters, we also mention Q5, Q6, and Q8 as other sub-problems which were the main concerns of this chapter. The role of the noise is considered and an optimization problem for the proposed signal model connecting to the underlying topology is proposed and solved.

5.5 Chapter Summary

We approached the main problem of this dissertation from the dictionary learning perspective. An algorithm is proposed to represent a set of multivariate measurements by a learned dictionary linked to a graph. As usual, it is also assumed that the graph signals are smooth with respect to the unknown underlying graph. The transformed graph in the dictionary domain, its relation to the signal smoothness and GFT have been discussed. Compared to the existing off-the-shelf algorithms, the proposed approach has a better performance, corroborated with some experimental results. Besides, the numerical results confirm that JGLSR is a practical algorithm to investigate temperature sensor readings and find how the weather propagates in different regions. Among the defined research questions in the first chapter, here, we approached the dictionary learning perspective in graph topology inference (Q3), the usage of atom coherence (Q4), connection of a kind of multivariate signal to the topology (Q5), dealing with denoising in the measurements (Q6), and formulation of a new optimization problem for graph topology inference (Q8).

Chapter 6

Adaptive Topology Learning

In this chapter, the concept of adaptive graph filters is introduced for online topology inference in data networks that are modeled as causal graph processes (CGP) or multivariate auto-regressive (MAR) processes. These models are time series associated with different variables, whose coefficients are the so-called graph filters. Given the data time series at different variables, we are interested in estimating these graph filters as well as learning the underlying graph weight matrix. The existing approaches focused on batch methods, assuming implicitly stationarity of the processes. Here, an adaptive method is proposed, based on the sequential arrival of new data. In addition to advantages in terms of complexity when compared to batch methods, these approaches also capture the dynamics of the topology. The experimental results support the performance of the proposed graphical adaptive filter algorithms in terms of signal recovery error and graph edge learning rate.

The huge amount of information collected anywhere and anytime in different environments has raised challenging issues of data storage, statistical processing, and information inferences. One possible way to cope with some of these issues is to leverage the underlying structure of data. Since the traditional approaches reviewed in sections 2.2.2.3 and 2.2.2.4 have limitations, the GSP framework motivated the topic of dynamic connectivity learning with data-driven approaches.

The focus of this chapter is on learning the directed topology from the given data set and it is an extension of the work in [138]. The directed structures are usually more specific and application-dependent. Some examples of the research works are [29, 30, 32–37, 123, 124]. Regarding the graph signal perspective, many of the directed graph signal processes are based on the cause-effect relationship. One of

the pioneering works in capturing such a relationship is Granger causality [125,126] which has been utilized in several applications [30, 127–134]. A time series $\mathbf{y}_i[k]$ Granger-causes the time series $\mathbf{y}_j[k]$ when knowledge about the past of $\mathbf{y}_i[k]$ highly improves $\mathbf{y}_j[k]$ prediction when compared to applying only the past of $\mathbf{y}_j[k]$ for predicting itself. The definition of Granger causality is general and some research works have proposed more specific definitions. Two of them are by Mei and Moura [27,28] and Bolstad et al. [31]. The former proposed the causal graph process (CGP) model and the latter focused on the multivariate auto-regressive (MAR) process model, also called vector auto-regressive (VAR). We review these signal models in section 6.1. The main problem is how to find the graph topology and recover the signal when the signal process follows these models.

In [27] and [31] the directed network topology estimation have been investigated, respectively, for CGP and MAR processes. They proposed batch algorithms that are not applicable for large/increasing data volume. Moreover, when the order of the underlying graph or the dimension of the graph signal is high, these algorithms are computationally expensive. The other limitation of the batch algorithm is when the processes are non-stationary. In this chapter, to overcome these issues, we propose an online learning method using the adaptive filter concept. Adaptive filters have self-adjusting capabilities. In other words, they are able to adjust their coefficients adapting to the input signal. One of the main applications of such filters is for signal-changing environments.

6.1 Graph Auto-Regressive Processes

The causal signal models assume that the current value of a time series at a given node is related to the previous values of both itself and other time series at other nodes. In what follows, two processes models are reviewed which investigate this concept for different scenarios.

6.1.1 Vector Auto-Regressive Signal Model

The Multivariate Auto-Regressive process is able to model the directed space-time dependencies as a specific implementation of the Granger causality. In the multiple time series analysis framework, MAR is sometimes equivalent to vector auto-regressive processes (VAR) and these two terms are used interchangeably. The MAR model has been used in many application, e.g. functional MRI analysis [135], classification of electroencephalographic signal [136], and so on [137].

In a MAR process of order M, an N-dimensional signal is a weighted superposition of M previous multivariate vectors added to an innovation noise. Mathematically speaking, a MAR process of order M is an LSI filter. It generates the signal at time instant k via a weighted combination of the previous M signals at all N nodes and adds an innovation noise $\mathbf{z}[k]$ as follows

$$\mathbf{x}[k] = \sum_{m=1}^{M} \mathbf{W}^{(m)} \mathbf{x}[k-m] + \mathbf{z}[k], \tag{6.1}$$

where matrices $\mathbf{W}^{(m)} \in \mathbb{R}^{N \times N}$ for $m = 1, \ldots, M$ are shift operators, like the one defined in 2.23. It can also be rewritten as the following element-wise expression

$$x_n[k] = \sum_{l:(n,l)\in\mathcal{E}} \sum_{m=1}^{M} w_{nl}^{(m)} x_l[k-m] + z_n[k], n = 1, \ldots, N \tag{6.2}$$

In many real-world applications, each time series is only affected by a subset of other time series. In other words, a node can not cause all the others. If $x_l[k]$ does not cause $x_n[k+m]$ for all $m \in \{1, \ldots, M\}$, it follows $w_{nl}^{(m)} = 0, \forall m$. Thus, a Sparse MAR Time series (SMART) is defined in [31] by considering a sparse set of edges.

6.1.2 Causal Graph Processes Model

Mei and Moura have applied the CGP concept to explain temperature propagation from one region to its neighborhoods [27, 28]. Thus, the current value of temperature at a given node is assumed to be related to the previous temperature values of the node itself and other neighboring nodes. Here, contrary to 6.1, a single weight matrix is utilized instead of multiple weight matrices which have the same sparse structure.

Let us assume that $\mathbf{x}[k] := [x_0[k],\, x_1[k], \ldots x_{N-1}[k]]^T$ is a random graph signal residing on $\mathcal{G}$ at time k. This graph process is a CGP if it is expressed as follows

$$\mathbf{x}[k] = \mathbf{z}[k] + \sum_{m=1}^{M} \Big(\sum_{j=0}^{m} h_{mj} \mathbf{W}^j \Big) \mathbf{x}[k-m], \tag{6.3}$$

where $h_{mj} \in \mathbb{C}$ and $\mathbf{z}[k]$ is an i.i.d. noise vector. Here, $\mathbf{h} = [h_{10}, h_{11}, \ldots h_{mj}, \ldots, h_{MM}]^T$ stores all of the filter coefficients h_{mj}, $1 \leq m \leq M$, $0 \leq j \leq m$ and thus we have $\sum_{j=0}^{m} h_{mj} \mathbf{W}^j \triangleq \mathbf{P}_m(\mathbf{W}, \mathbf{h})$. By applying this model, the data is propagated in the rate of one graph shift per sampling period. According to the

definition 2.25, $\mathbf{P}_m(\mathbf{W}, \mathbf{h})$ is an LSI filter of order M. Since the weight matrix is not an identity matrix, we have $\mathbf{P}_1(\mathbf{W}, \mathbf{h}) \neq c\mathbf{I}_N$, for any constant $c \in \mathbb{R}$.

6.2 Learning Topology from AR Processes

In what follows, we are given a set of noisy observations from AR processes. In section 6.2.1, the batch approach for graph learning from MAR processes is reviewed which has been introduced in [31]. In section 6.2.2, a batch approach to learn the underlying graph from CGP [27] is reviewed. These two methods require all data in advance. However, in the last subsection 6.2.3, we propose to jointly recover the topology and graph signals online. We only investigate this adaptive method for CGP here, since a similar procedure can be applied for the MAR process.

6.2.1 Batch Mode for MAR Processes

Suppose that a set of noisy observations from a MAR process is given. Considering 6.1, the problem of graph topology inference is estimating $\mathbf{W}^{(m)}$, $m = 1, \ldots, M$. If $\mathbf{w}_{nl} \triangleq \left[w_{nl}^{(1)}, w_{nl}^{(2)}, \ldots, w_{nl}^{(M)}\right]^T$, the weight matrices can be estimated as follows [31]

$$\begin{aligned}\{\hat{\mathbf{W}}^{(m)}\}_{m=1}^{M} = \underset{\{\mathbf{W}^{(m)}\}_{m=1}^{M}}{\operatorname{argmin}} \sum_{k=M+1}^{K} \left\| \mathbf{x}[k] - \sum_{m=1}^{M} \mathbf{W}^{(m)} \mathbf{x}[k-m] \right\|_2^2 \\ + \lambda \sum_{n=1}^{N} \sum_{l=1}^{N} \mathbb{1}\Big(\left\| \mathbf{w}_{nl} \right\|_1 \Big),\end{aligned} \tag{6.4}$$

where $\lambda > 0$ is a regularization parameter and $\mathbb{1}(.)$ is the indicator function which outputs 0 and 1 for zero and non-zero inputs, respectively. The first term of (6.4) encourages data fidelity and the ℓ_1-norm promotes sparse solutions. The indicator function enforces the group sparsity. The group sparsity means that if all the elements of $\mathbf{w}_{nl}$ are zero, the node l does not cause the node n for all the filter taps m which causes a more sparse solution. When there is even one non-zero edge $w_{nl}^{(m)}$ between the node n and the node l, it confirms that the node n is affected by the node l. Thus, the indicator function outputs one, adding to the cost due to an additional non-zero element in the filter weight matrices. Since (6.4) is non-convex, by following the group Lasso minimization [31], it is rewritten as follows

$$\{\hat{\mathbf{w}}_n\}_{n=1}^{N} = \underset{\{\mathbf{w}_n\}_{n=1}^{N}}{\operatorname{argmin}} \sum_{n=1}^{N} \Big(\sum_{k=M+1}^{K} \big(x_n[k] - \mathbf{g}^T[k] \mathbf{w}_n \big)^2 + \lambda \sum_{l=1}^{N} \left\| \mathbf{w}_{nl} \right\|_2 \Big), \tag{6.5}$$

where

$$\mathbf{w}_n = \left[\mathbf{w}_{n,1}^T,\, \mathbf{w}_{n,2}^T, \ldots, \mathbf{w}_{n,N}^T\right]^T, \tag{6.6}$$

$$\mathbf{g}[k-1] \triangleq \text{vec}\Big(\Big[\mathbf{x}[k-1], \mathbf{x}[k-2], \ldots, \mathbf{x}[k-M]\Big]^T \Big). \tag{6.7}$$

Due to the separability of the cost function across $\mathbf{w}_n$, (6.5) is solved for each n as below [31]

$$\hat{\mathbf{w}}_n = \underset{\mathbf{w}_n}{\text{argmin}} \sum_{k=M+1}^{K} \left(x_n[k] - \mathbf{g}^T[k]\mathbf{w}_n\right)^2 + \lambda \sum_{l=1}^{N} \|\mathbf{w}_{nl}\|_2 . \tag{6.8}$$

6.2.2 Batch Mode for CGP

Suppose that a set of graph signals from a CGP is given and the weight matrix is desired. Thus, the aim is to estimate $\mathbf{W}$ or $\mathbf{P}_m(\mathbf{W}, \mathbf{h})$ for $m = 1, \ldots, M$ when the signal model is represented by 6.3. To estimate $\mathbf{W}$ and $\mathbf{h}$, a minimization problem is proposed in [27] assuming that $\mathbf{z}[k]$ is Gaussian and both $\mathbf{W}$ and $\mathbf{h}$ are sparse. This minimization problem is as follows

$$\begin{aligned}(\widehat{\mathbf{W}}, \hat{\mathbf{h}}) = \underset{\mathbf{W},\mathbf{h}}{\text{argmin}} \frac{1}{2} \sum_{k=M+1}^{K} \left\| \mathbf{x}[k] - \sum_{m=1}^{M} \mathbf{P}_m(\mathbf{W}, \mathbf{h})\mathbf{x}[k-m] \right\|_2^2 \\ + \lambda_1 \left\|\text{vec}(\mathbf{W})\right\|_1 + \lambda_2 \left\|\mathbf{h}\right\|_1 .\end{aligned} \tag{6.9}$$

where $\lambda_1,\ \lambda_2$ are regularization parameters. This problem is non-convex due to $\mathbf{W}^j$ for $j > 1$. In [27], it is proposed to reformulate this problem based on a matrix variable $\mathbf{R} = [\mathbf{R}_1 \mid \mathbf{R}_2 \mid ... \mid \mathbf{R}_M]$ as below

$$\begin{aligned}\hat{\mathbf{R}} = \underset{\mathbf{R}}{\text{argmin}} \frac{1}{2} \sum_{k=M+1}^{K} \left\| \mathbf{x}[k] - \sum_{m=1}^{M} \mathbf{R}_m\mathbf{x}[k-m] \right\|_2^2 \\ + \lambda_1 \left\|\text{vec}(\mathbf{R})\right\|_1 + \lambda_3 \sum_{m \neq j} \left\|[\mathbf{R}_m,\, \mathbf{R}_j]\right\|_F^2 ,\end{aligned} \tag{6.10}$$

where

$$[\mathbf{R}_m,\, \mathbf{R}_j] = \mathbf{R}_m\mathbf{R}_j - \mathbf{R}_j\mathbf{R}_m,\ \forall m, j. \tag{6.11}$$

Since $\mathbf{R}_m \triangleq \mathbf{P}_m(\mathbf{W}, \mathbf{h})$ is polynomial and shift-invariant, it satisfies the mutual commutativity, i.e. (6.11). To solve (6.10), a Block Coordinate Descent (BCD) method is proposed in [27]. In the iteration τ, each sub-problem of BCD algorithm aims to estimate $\hat{\mathbf{R}}_m^\tau$ while the other $\hat{\mathbf{R}}_l, l \neq m$ are kept fixed with their latest

updates

$$\begin{aligned}\hat{\mathbf{R}}_m^{\tau} = \underset{\mathbf{R}_m}{\operatorname{argmin}} \frac{1}{2} \sum_{k=M+1}^{K} \|\mathbf{b}_{-m}[k] - \mathbf{R}_m \mathbf{x}[k-m]\|_2^2 \\ + \lambda_{R_m} \|\text{vec}(\mathbf{R}_m)\|_1 + \lambda_3 \sum_{l \neq m} \Big\| [\mathbf{R}_m,\, \hat{\mathbf{R}}_l] \Big\|_F^2 .\end{aligned} \tag{6.12}$$

where $\hat{\mathbf{R}}_l = \hat{\mathbf{R}}_l^{\tau}$ for $l < i$ and $\hat{\mathbf{R}}_l = \hat{\mathbf{R}}_l^{\tau-1}$ for $l > i$. The i'th polynomial filter and the commutativity regularization parameters are λ_{R_i} and λ_3, respectively. Also, $\mathbf{b}_{-m}[k]$ is given as follows

$$\mathbf{b}_{-m}[k] = \mathbf{x}[k] - \Big(\sum_{l=1}^{m-1} \hat{\mathbf{R}}_l^{\tau} \mathbf{x}[k-l] + \sum_{l=m+1}^{M} \hat{\mathbf{R}}_l^{\tau-1} \mathbf{x}[k-l] \Big). \tag{6.13}$$

The weight matrix is either estimated by $\hat{\mathbf{W}} = \hat{\mathbf{R}}_1$ or as follows

$$\widehat{\mathbf{W}} = \min_{\mathbf{W}} \Big\| \hat{\mathbf{R}}_1 - \mathbf{W} \Big\|_2^2 + \lambda_1 \|\text{vec}(\mathbf{W})\|_1 + \lambda_3 \sum_{m=2}^{M} \Big\| [\mathbf{W},\, \hat{\mathbf{R}}_m] \Big\|_F^2 \tag{6.14}$$

and then we have

$$\hat{\mathbf{h}}_m = \underset{\mathbf{h}_m}{\operatorname{argmin}} \frac{1}{2} \Big\| \text{vec}(\hat{\mathbf{R}}_{\mathbf{1}}) - \boldsymbol{\Upsilon}_m \mathbf{h}_m \Big\|_2^2 + \lambda_2 \|\mathbf{h}_m\|_1 , \tag{6.15}$$

where $\boldsymbol{\Upsilon}_m = \Big[\text{vec}(\mathbf{I}_N) \mid \text{vec}(\widehat{\mathbf{W}}) \mid \ldots \mid \text{vec}(\widehat{\mathbf{W}}^m) \Big]$ and $\mathbf{h}_m = \big[h_{m0},\, h_{m1},\, \ldots h_{mm} \big]^T$.

For a graph with a relatively high order, the matrix of the graph filter polynomial is large, which causes the batch algorithm to have high computational complexity. Besides, for non-stationary processes, it is necessary to design an algorithm to be able to re-estimate the graph filter multiple times, adapted to the input signal. In real-time applications, the new set of signal samples are received sequentially over time and hence all the data is not at hand in advance. For such cases, there is a need to have an online algorithm to extract data structure recursively.

6.2.3 Adaptive Graph Filtering

We investigate a recursive algorithm to adapt the graph filters continuously once a new sample is provided. This adaptive filter is not only capable of graph weight matrix learning based on more recent observations, but also it recovers the graph signals from noisy measurements.

6.2.3.1 Mathematical Analysis

Considering Kronecker product properties, it is known that if $\mathbf{A}_1, \mathbf{A}_2,$ and $\mathbf{A}_3$ are three matrices with appropriate dimensions,

$$\text{vec}(\mathbf{A}_1\mathbf{A}_2\mathbf{A}_3) = (\mathbf{A}_3 \otimes \mathbf{A}_1)\text{vec}(\mathbf{A}_2). \tag{6.16}$$

By some manipulations given in [138], (6.10) can be rewritten in the following vector form

$$\begin{aligned}\hat{\mathbf{r}}_m^{\tau} = \underset{\mathbf{r}_m}{\text{argmin}}\ \frac{1}{2}\sum_{k=M+1}^{K} \left\|\mathbf{b}_{-m}[k] - \big(\mathbf{x}[k-m]^T \otimes \mathbf{I}_N\big)\mathbf{r}_m\right\|_2^2 \\ + \lambda_{r_m}\left\|\mathbf{r}_m\right\|_1 + \lambda_3 \sum_{l\neq m}\left\|[\mathbf{R}_m,\, \hat{\mathbf{R}}_l]\right\|_F^2,\end{aligned} \tag{6.17}$$

where $\mathbf{r}_m = \text{vec}(\mathbf{R}_m)$. We also have

$$\text{vec}(\mathbf{R}_m\mathbf{R}_l - \mathbf{R}_l\mathbf{R}_m) = \Big(\big(\mathbf{R}_l \otimes \mathbf{I}_N\big) - \big(\mathbf{I}_N \otimes \mathbf{R}_l\big)\Big)\text{vec}(\mathbf{R}_m). \tag{6.18}$$

$$\sum_{l\neq m}\|[\mathbf{R}_m,\, \mathbf{R}_l]\|_F^2 = \sum_{l\neq i}\left\|\Big(\big(\mathbf{R}_l \otimes \mathbf{I}_N\big) - \big(\mathbf{I}_N \otimes \mathbf{R}_l\big)\Big)\mathbf{r}_m\right\|_2^2. \tag{6.19}$$

Substituting (6.19) in (6.17) and applying the forgetting factor β[1], the objective function is adopted from [138] as below

$$\begin{aligned}J_K(\mathbf{r}_m^{\tau}) =& \frac{1}{2}\sum_{k=M+1}^{K} \beta^{K-k}\left\|\mathbf{b}_{-m}[k] - \big(\mathbf{x}[k-m]^T \otimes \mathbf{I}_N\big)\mathbf{r}_m^{\tau}\right\|_2^2 \\ &+ \lambda_{r_m}\left\|\mathbf{r}_m^{\tau}\right\|_1 + \lambda_3 \sum_{l\neq m}\left\|\mathbf{\Gamma}_l\mathbf{r}_m^{\tau}\right\|_2^2.\end{aligned} \tag{6.20}$$

where

$$\mathbf{\Gamma}_l = \big(\mathbf{R}_l^T \otimes \mathbf{I}_N\big) - \big(\mathbf{I}_N \otimes \mathbf{R}_l\big). \tag{6.21}$$

Defining $\mathbf{V}[k] := \mathbf{x}[k-m]^T \otimes \mathbf{I}_N$ and applying some simple manipulations

[1] in designing recursive filters, we can consider the more recent signal values as more important than that of the previous ones. In this respect, sometimes a forgetting factor $0 < \beta < 1$ is defined to reduce the effect of the older data.

leads to the following recursion

$$J_K\big(\mathbf{r}_m^\tau\big) = \frac{1}{2}(\mathbf{r}_m^\tau)^T\Big(\lambda_3 \sum_{l\neq m} \mathbf{\Gamma}_l^T\mathbf{\Gamma}_l + \sum_{k=M+1}^{K} \beta^{K-k}\mathbf{V}[k]^T\mathbf{V}[k]\Big)\mathbf{r}_m^\tau - (\mathbf{r}_m^\tau)^T \sum_{k=M+1}^{K} \beta^{K-k}\mathbf{V}[k]^T\mathbf{b}_{-m}[k] + \frac{1}{2}\sum_{k=M+1}^{K} \beta^{K-k}\mathbf{x}_k^T\mathbf{b}_{-m}[k] + \lambda_{r_m}\left\|\mathbf{r}_m^\tau\right\|_1, \tag{6.22}$$

The ℓ_1-norm term is convex but not differentiable. Therefore, solving this problem by the standard gradient descent is not straightforward. However, it is possible to apply a subgradient instead. The vector $\mathbf{u}$ is a subgradient of $g : \mathbf{R}^n \to \mathbf{R}$ at $\mathbf{y} \in \operatorname{dom} g$ when $g(\mathbf{y}') \geq g(\mathbf{y}) + \mathbf{u}^T(\mathbf{y}' - \mathbf{y}), \forall \mathbf{y}' \in \operatorname{dom} g$. The set of all subgradients of g at $\mathbf{y}$ is called the subdifferential. A function g is called subdifferentiable at $\mathbf{y}$ if there exists at least one subgradient at $\mathbf{y}$. A point $\mathbf{y}^* \in \mathbb{R}^N$ minimizes the non differentiable convex function g if and only if $\mathbf{0}$ is a subgradient of g at $\mathbf{y}$ [139].

A subgradient of the objective function in (6.22) is given in [138]

$$\nabla^s J_K\big(\mathbf{r}_i^\tau\big) = \big(\mathbf{Q}[K] + \lambda_3 \sum_{l\neq m} \mathbf{\Gamma}_l^T\mathbf{\Gamma}_l\big)\mathbf{r}_m^\tau - \mathbf{q}[K] + \lambda_{r_m}\operatorname{sign}\big(\mathbf{r}_m^\tau\big), \tag{6.23}$$

where

$$\mathbf{Q}[K] = \sum_{k=M+1}^{K} \beta^{K-k}\mathbf{V}[k]^T\mathbf{V}[k], \tag{6.24}$$

and

$$\mathbf{q}[K] = \sum_{k=M+1}^{K} \beta^{K-k}\mathbf{V}[k]^T\mathbf{b}_{-m}[k] \tag{6.25}$$

By some simple manipulations, we have

$$\mathbf{Q}[K] = \beta\mathbf{Q}[K-1] + \mathbf{V}[K]^T\mathbf{V}[K] \tag{6.26}$$

$$\mathbf{q}[K] = \beta\mathbf{q}[K-1] + \mathbf{V}[K]^T\mathbf{b}_{-m}[K]. \tag{6.27}$$

Applying the standard gradient descent leads to

$$(\mathbf{r}_m^\tau)_t = (\mathbf{r}_m^\tau)_{t-1} - \alpha_t\nabla^s J_K\big((\mathbf{r}_m^\tau)_{t-1}\big). \tag{6.28}$$

where α_t is a step-size that guarantees the convergence and t is the iteration index.

Algorithm 6.1 summarizes all steps for the adaptive graph filter for CGP.

6.2.3.2 Computational Complexity Analysis

As we discussed in [138], finding the solution of (6.12) for any newly provided set of time series $\mathbf{x}[k]$ is highly time consuming due to the worst-case complexity of $\mathcal{O}(M^2N^3 + KMN^2)$ [27]. Usually we have $K \gg N \gg M$ and thus the computational cost may be approximated as $\mathcal{O}\big(KMN^2 \min(K, MN^2)\big)$. Besides, K grows linearly while M are kept constant and N are fixed by the nature of the graph signal dimension, so we have $\min(K, MN^2) \simeq MN^2$ in reality. Therefore, the computational complexity may be approximated by $\mathcal{O}(KM^2N^4)$. Since this cost is directly proportional to K and hence increases with time, solving (6.12) in batch mode is prohibitive in terms of computational complexity and storage demands. Roughly speaking, the sample size K increases by time, and for each time the batch mode must run which is computationally expensive.

The computational cost of the proposed algorithm is dominated by the processing of (6.23), upper bounded by $\mathcal{O}(M^2N^4)$ due to the matrix-vector multiplication of the first term. Hence, it is identical to the batch mode for $K = 1$, which is supporting the concept of the recursive method. Intuitively, the GRLS only runs for the recently given set of observations.

6.2.3.3 Experimental Results

An Erdős Rényi graph is generated with $N = 25$ and edges weights are extracted from a Gaussian pdf $\mathcal{N}(0, 1)$. After deleting all the edges excepts the ones with a

Algorithm 6.1: Topology learning via adaptive graph filtering, adopted from [138].

Input: $\{\mathbf{x}[k]\}_{k=1}^{K}, \lambda_{r_m}, \lambda_2, \lambda_3, t_{\max}, \{\alpha_t\}_{t=1}^{t_{\max}}, \tau_{\max}, M$
Initialize: $\hat{\mathbf{R}}_m = \mathbf{I}_N$ for $m = 1 : M$
while $\mathbf{x}[k], k > M$ **do**
  Update the polynomial filters
  for $\tau = 1 : \tau_{max}$ **do**
    for *m=1:M* **do**
      for $t = 1 : t_{max}$ **do**
        a) find $\nabla^s J_K\big(\mathbf{r}_i^\tau\big)$ via (6.23)
        b) update $(\mathbf{r}_i^\tau)_t$ via (6.28)
      end
    end
  end
  Find the weight matrix $\widehat{\mathbf{W}} = \hat{\mathbf{R}}_1$ and filter coefficients.
end

Table 6.1: Performance of GRLS filter for different number of time series, taken from [138].

	$K = 75$	$K = 100$	$K = 125$	$K = 150$
NMSD	0.0035	0.0033	0.0032	0.0031
Recall	0.66	0.76	0.78	0.82

weight in the interval [1.6, 1.8], the edge probability is 0.04. The edges are soft thresholded by 1.5 and normalized by 1.5 times of the largest eigenvalue of the weight matrix. Then, by setting $M = 3$ and generating the coefficients h_{mj} for $m = \{2, 3\}, 0 \leq j \leq m$ sparsely from $2^{m+j+1}h_{mj} \sim \mathcal{U}(-1, -0.45) + \mathcal{U}(0.45, 1)$, a stable system is guaranteed [27]. The first graph signals $\mathbf{x}[k], k = 1, \ldots, M$ are generated from a random Gaussian distribution with zero mean and unit variance and for $M + 1 < k \leq K$, $K = 150$, all the rest is generated according to (6.3).

Here, we define the normalized mean square deviation (NMSD) as follows

$$\text{NMSD} = \frac{\left\|\mathbf{W} - \hat{\mathbf{W}}\right\|_F^2}{N^2} \tag{6.29}$$

which is 0.0029 for the batch mode and it varies for GRLS implementation depending on the number of observations, noted in Table 6.1. When the number of time samples is increasing, the reconstruction error decreases, and then the NMSD of the GRLS filter converges to the NMSD of the batch method. The other performance measure is the "Recall" factor which is the ratio of the number of correct recovered edges to the number of true edges in the ground-truth graph. Here, we have the recall factor of 87 percent for the batch mode while it varies between 66 to 82 percent for GRLS.

In the next simulation, we examined GRLS for a large number of measurements. The results are averaged over 50 trials and the maximum K is set to 1000. The NMSD and recall for batch mode have been 0.0039 and 91 percent, respectively. Figure 6.1 shows the performance measures for the proposed adaptive graph learning algorithm.

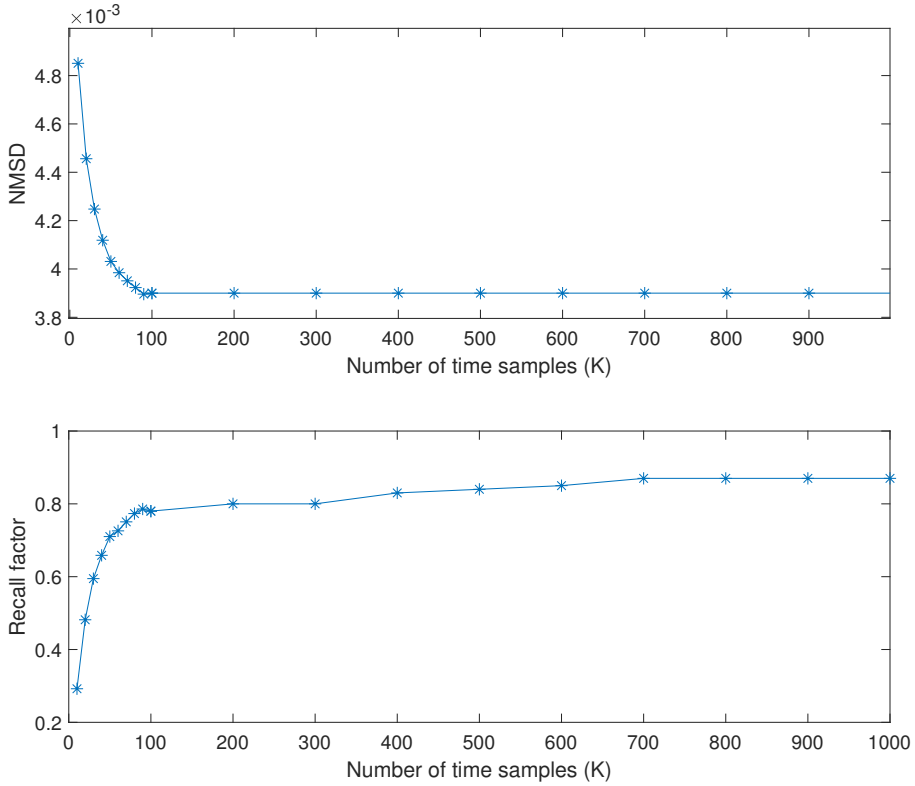


Figure 6.1: NMSD and Recall for $N = 25$ and $M = 2$, taken from [138].

6.3 Discussion

The main difference between this chapter and the previous three ones was its concentration on the directed topology and linked signal processes. Since some important multivariate signal models like MAR and CGP have been investigated in this chapter, the results can answer Q2 and Q9 of Sec. 1.2. Also, the first question among the research questions of this thesis which concerns the usage of conventional adaptive filters to the graph signal processing framework was investigated in this chapter. Besides, like the previous chapters, the general Q5, Q6, and Q8 which directly aimed at answering the main question of this thesis, was approached in this chapter for the directed topology learning.

6.4 Chapter Summary

In this chapter, it is assumed that a set of multivariate signals over graph is given and the desired goal is to learn the underlying topology and recover the graph signals from the noisy measurements. Since the graph signals are generated from a CGP

or a MAR process, the estimated topology is directed and captures the cause-effect relationships among entities of the network. The proposed recursive least square filter is capable of graph topology learning in an online manner while the classical batch mode algorithm works when we have all the data in advance. Since the proposed method is recursive and updates estimates online, it is able to keep track of changes in the topology by feeding the sequential arrival of new data. Thus, it also suits the non-stationary processes and has low computational complexity. Regarding the defined research questions in the first chapter, here, we showed how the conventional adaptive filter can be applied for the graph topology inference (Q1) and online structure recovery (Q2). Besides, we discussed the connection of a kind of multivariate causal process to the topology (Q5), the contaminated measurements with noise (Q6), the formulation of an optimization problem for graph topology inference (Q8), and its fast implementation (Q9).

Chapter 7

Conclusions and Future Works

This chapter is organized into two sections. In the first section, we summarize the dissertation work and the main contributions. The second section points out a few potential research directions relevant to the topics addressed in this dissertation.

7.1 Conclusion

The real-world applications generate rapidly growing volumes of structured data, e.g. brain-computer interface measurements, social networks activities, gene network data, patient records of healthcare systems, and financial data. Storing and analyzing these data sets are easier when the underlying data structure is considered. The emerging field of graph signal processing (GSP) provides the analysis of large data sets via graph theory tools, where each graph vertex represents an entity of the system. A sequence of data is generated by each entity and hence a data matrix including the information over time and space is provided. If the underlying graph is known, the system can be analyzed from different perspectives. For example, the spectral analysis or the Fourier transform can be done and some signal characteristics are investigated in this way.

Sometimes the graph topology is not known in advance. In such cases, the desired goal is to estimate the underlying topology using a set of graph signals. This was the main focus of this dissertation. In the non-GSP-based methods, the graph topology learning has been limited to find the connections among high dimensional signals. However, in this thesis, following the GSP framework, we focused on the global behavior of signals over graphs instead of pairwise relationships.

For applications dealing with cause-effect relationships among nodes, the topol-

ogy is directed and for some others, especially the ones corresponding to the Gaussian graph signals, the learned topology is undirected. We reviewed three main signal processes here as multivariate auto-regressive (MAR), causal graph processes (CGP), and Gaussian Markov random field (GMRF) processes. For the first two ones, there have been some approaches in the literature of graph topology learning, but they were in the batch mode and thus they are slow. We proposed an adaptive filter specifically designed for graph signals. The main advantages of such filters are that they can be implemented online and are capable of adapting to the newly arrived signals. Therefore, the re-estimation of the learned topology is possible based on the recently provided measurements.

For the GMRF signal models, we proposed a topology learning algorithm based on the Bayesian inference framework. A factor analysis model was used to decompose the graph signals based on the eigenvectors of the graph Laplacian matrix. In this way, we fully connected the given data set to the underlying topology. A fast implementation method was also proposed which outperforms the conventional approaches.

Moreover, for the general graph signal model, we designed a dictionary learning algorithm to learn the atoms based on the graph topology. In this respect, we jointly recovered the dictionary atoms, the sparse codes, and the underlying topology. Another contribution of this dissertation for general graph signal models is to learn the graph topology from a data set that is noisy and contaminated with outliers. We showed that the proposed method is applicable in a real-world scenario, e.g. learning an information network from stock market share prices data.

7.2 Future Works

Based on the research results achieved in this thesis work, a few potential topics can be identified. In brief, the following research directions are foreseen.

- Among the adaptive filters, the easiest to implement with the least computational resources needed is the least mean square (LMS) filter. The LMS works on the current data and the one which comes in. The RLS, which is more computationally intensive, works on all data gathered till now and is a sequential way to solve the Wiener filter. Thus, if a very simple and fast online solution is required, we can think of designing an LMS filter for graph signals. Especially for non-stationary data, the LMS filter is a good candidate.

- In the CGP model discussed in chapter 6 and also in [27], the adjacency

matrix is used. This whole framework can be more general and any valid graph shift operator can be used to construct the polynomials. For example, the polynomials can be constructed by the graph Laplacian and hence capture an undirected graph.

- The subgradient descent method used to solve (6.22) may be a bit slow when compared to other off-the-shelf solvers for ℓ_1-ℓ_2 problems. One of the future works is to focus on solving this problem with different methods and find the convergence rate and implementation complexity for different kinds of graph structures. For example, the iterative shrinkage-thresholding algorithm (ISTA) (or similar ones [140–142]) is an extension of the gradient algorithms and is attractive here due to its simplicity and also it is a good candidate to solve large-scale problems. Another method for further investigation is Fast ISTA (FISTA) [143], which has the same simplicity as ISTA, while its global rate of convergence is significantly better.

- About the approach discussed in chapter 4, the followings are of future research interest.

  - we would like to see more discussions on the selection of the regularization parameters and constants. Especially, c_1 is of interest. For example, if it is infinity, the solution is not meaningful. What is the intuition for other values?

  - Although the simulation results show that the RTLG converges, the analytical proof of convergence is not presented here and left for future research work.

  - A very interesting topic is to design a robust method for non-stationary processes. This can be helpful for brain activity signal analysis. In that application, we have a changing topology for a subject in different conditions where the external outliers are playing an important role.

- In chapter 5, the proposed method focuses on topology learning and signal representation. An interesting area of research is to investigate the transformed graph properties. How the relation between graph signals and the underlying structure can be projected to the relation of sparse codes and the transformed graph in the dictionary domain. For example, what happens to the smoothness concept in the dictionary domain, or is it possible to propose a smoothness concept in the dictionary domain directly instead of using the same measure of $\text{Tr}(\mathbf{X}^T\tilde{\mathbf{L}}\mathbf{X})$ in (5.4)?

- In chapter 3, it is assumed that the graph topology is fixed and does not depend on k. Extending the problem to the one that estimates a changing graph is left for future work. In this respect, we can investigate an online method to update the dictionary atoms based on the recently provided signals.

REFERENCES

[1] T. Paoletti, "Leonard euler's solution to the konigsberg bridge problem," *Convergence*, 2011.

[2] J. A. Deri, F. Franchetti, and J. M. Moura, "Big data computation of taxi movement in new york city," in *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, 2016, pp. 2616–2625.

[3] 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 Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, May 2013.

[4] M. Ramezani Mayiami, M. Hajimirsadeghi, K. Skretting, X. Dong, R. S. Blum, and H. V. Poor, "Bayesian topology learning and noise removal from network data," *Discover Internet of Things*, vol. 1, no. 1, pp. 1–21, 2021.

[5] N. Perraudin, J. Paratte, D. Shuman, L. Martin, V. Kalofolias, P. Vandergheynst, and D. K. Hammond, "GSPBOX: A toolbox for signal processing on graphs," *ArXiv e-prints*, Aug. 2014.

[6] M. Ramezani-Mayiami and K. Skretting, "Robust graph topology learning and application in stock market inference," in *2019 IEEE International Conference on Signal and Image Processing Applications (ICSIPA)*. IEEE, 2019, pp. 240–244.

[7] S. Zhang, X. Jiang, W. Zhang, T. Zhang, H. Chen, Y. Zhao, J. Lv, L. Guo, B. R. Howell, M. M. Sanchez *et al.*, "Joint representation of connectome-scale structural and functional profiles for identification of consistent cortical landmarks in macaque brain," *Brain imaging and behavior*, vol. 13, no. 5, pp. 1427–1443, 2019.

[8] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[9] P. Hagmann, L. Cammoun, X. Gigandet, R. Meuli, C. J. Honey, V. J. Wedeen, and O. Sporns, "Mapping the structural core of human cerebral cortex," *PLoS biology*, vol. 6, no. 7, p. e159, 2008.

[10] C. Apté, F. Damerau, and S. M. Weiss, "Automated learning of decision rules for text categorization," *ACM Transactions on Information Systems (TOIS)*, vol. 12, no. 3, pp. 233–251, 1994.

[11] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovačević, "Adaptive graph filtering: Multiresolution classification on graphs," in *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 427–430.

[12] S. Chen, A. Sandryhaila, G. Lederman, Z. Wang, J. M. Moura, P. Rizzo, J. Bielak, J. H. Garrett, and J. Kovačević, "Signal inpainting on graphs via total variation minimization," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 8267–8271.

[13] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, "Discrete signal processing on graphs: Sampling theory," *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6510–6523, Dec 2015.

[14] S. Chen, A. Sandryhaila, J. M. Moura, and J. Kovačević, "Signal recovery on graphs: Variation minimization," *IEEE Transactions on Signal Processing*, vol. 63, no. 17, pp. 4609–4624, 2015.

[15] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Processing Magazine*, vol. 31, no. 5, pp. 80–90, 2014.

[16] ——, "Discrete signal processing on graphs," *IEEE transactions on signal processing*, vol. 61, no. 7, pp. 1644–1656, 2013.

[17] D. K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[18] X. Zhu and M. Rabbat, "Graph spectral compressed sensing for sensor networks," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 2865–2868.

[19] X. Dong, "Multi-view signal processing and learning on graphs," EPFL, Tech. Rep., 2014.

[20] A. Loukas, A. Simonetto, and G. Leus, "Distributed autoregressive moving average graph filters," *IEEE Signal Processing Letters*, vol. 22, no. 11, pp. 1931–1935, Nov 2015.

[21] A. Sandryhaila, S. Kar, and J. M. F. Moura, "Finite-time distributed consensus through graph filters," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, May 2014, pp. 1080–1084.

[22] W. Xu, E. Mallada, and A. Tang, "Compressive sensing over graphs," in *2011 Proceedings IEEE INFOCOM*. IEEE, 2011, pp. 2087–2095.

[23] A. Gadde and A. Ortega, "A probabilistic interpretation of sampling theory of graph signals," in *2015 IEEE international conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2015, pp. 3257–3261.

[24] S. Chen, "Data science with graphs: A signal processing perspective," Ph.D. dissertation, figshare, 2016.

[25] A. Sandryhaila and J. M. Moura, "Classification via regularization on graphs," in *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 495–498.

[26] ——, "Discrete signal processing on graphs: Frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042–3054, 2014.

[27] J. Mei and J. M. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077–2092, 2017.

[28] J. Mei and J. M. F. Moura, "Signal processing on graphs: Estimating the structure of a graph," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, April 2015, pp. 5495–5499.

[29] K. J. Friston, "Functional and effective connectivity in neuroimaging: a synthesis," *Human brain mapping*, vol. 2, no. 1-2, pp. 56–78, 1994.

[30] 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, 2003.

[31] A. Bolstad, B. D. V. Veen, and R. Nowak, "Causal network inference via group sparse regularization," *IEEE Transactions on Signal Processing*, vol. 59, no. 6, pp. 2628–2641, June 2011.

[32] Y. Shen, B. Baingana, and G. B. Giannakis, "Kernel-based structural equation models for topology identification of directed networks," *IEEE Transactions on Signal Processing*, vol. 65, no. 10, pp. 2503–2516, May 2017.

[33] L. A. Baccalá and K. Sameshima, "Partial directed coherence: a new concept in neural structure determination," *Biological cybernetics*, vol. 84, no. 6, pp. 463–474, 2001.

[34] J. Songsiri and L. Vandenberghe, "Topology selection in graphical models of autoregressive processes," *Journal of Machine Learning Research*, vol. 11, no. Oct, pp. 2671–2705, 2010.

[35] Y. Shen, B. Baingana, and G. B. Giannakis, "Nonlinear structural vector autoregressive models for inferring effective brain network connectivity," *arXiv preprint arXiv:1610.06551*, 2016.

[36] B. Baingana and G. B. Giannakis, "Tracking switched dynamic network topologies from information cascades," *IEEE Transactions on Signal Processing*, vol. 65, no. 4, pp. 985–997, 2016.

[37] P. A. Traganitis, Y. Shen, and G. B. Giannakis, "Network topology inference via elastic net structural equation models," in *Proc. IEEE 25th European Signal Processing Conference (EUSIPCO)*. IEEE, 2017, pp. 146–150.

[38] A. P. Dempster, "Covariance selection," *Biometrics*, pp. 157–175, 1972.

[39] O. Banerjee, L. E. Ghaoui, and A. d'Aspremont, "Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data," *Journal of Machine Learning Research*, vol. 9, no. Mar, pp. 485–516, 2008.

[40] B. Lake and J. Tenenbaum, "Discovering structure by learning sparse graph," in *Proc. 32nd Annual Meeting of the Cognitive Science Society*, 2010, pp. 6160–6173.

[41] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian matrix in smooth graph signal representations," *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, Dec 2016.

[42] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical LASSO," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[43] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. 19th International Conference of Artificial Intelligence and Statistics, AISTATS, Cadiz*, 2016, pp. 920–929.

[44] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 467–483, Sept 2017.

[45] M. Yuan and Y. Lin, "Model selection and estimation in the gaussian graphical model," *Biometrika*, vol. 94, no. 1, pp. 19–35, 2007.

[46] K. Scheinberg and I. Rish, "Learning sparse gaussian markov networks using a greedy coordinate ascent approach," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2010, pp. 196–212.

[47] R. Shafipour, S. Segarra, A. G. Marques, and G. Mateos, "Network topology inference from non-stationary graph signals," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 5870–5874.

[48] ——, "Identifying the topology of undirected networks from diffused non-stationary graph signals," *arXiv preprint arXiv:1801.03862*, 2018.

[49] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of graph diffusion processes from observations of stationary signals," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 4, no. 3, pp. 481–496, Sep. 2018.

[50] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 484–499, Sep. 2017.

[51] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from filtered signals: Graph system and diffusion kernel identification," *IEEE Transactions on Signal and Information Processing over Networks*, 2018.

[52] H. Ma, H. Yang, M. R. Lyu, and I. King, "Mining social networks using heat diffusion processes for marketing candidates selection," in *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 2008, pp. 233–242.

[53] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under Laplacian and structural constraints," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 6, pp. 825–841, 2017.

[54] Y. Yankelevsky and M. Elad, "Dual graph regularized dictionary learning," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 2, no. 4, pp. 611–624, 2016.

[55] L. Euler, "Solutio problematis ad geometriam situs pertinentis," *Commentarii academiae scientiarum Petropolitanae*, pp. 128–140, 1741.

[56] J. J. Sylvester, "Chemistry and algebra," 1878.

[57] J. A. Bondy, U. S. R. Murty *et al.*, *Graph theory with applications*. Macmillan London, 1976, vol. 290.

[58] D. B. West *et al.*, *Introduction to graph theory*. Prentice hall Upper Saddle River, NJ, 1996, vol. 2.

[59] W. T. Tutte *et al.*, *Graph theory as I have known it*. Oxford University Press, 1998, no. 11.

[60] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[61] E. D. Kolaczyk, *Statistical analysis of network data: methods and models*. Springer Science & Business Media, 2009.

[62] S. L. Lauritzen, *Graphical models*. Clarendon Press, 1996, vol. 17.

[63] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Laplacian matrix learning for smooth graph signal representation," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3736–3740.

[64] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 787–807, 2018.

[65] S. Sardellitti, S. Barbarossa, and P. Di Lorenzo, "Graph topology inference based on sparsifying transform learning," *IEEE Transactions on Signal Processing*, vol. 67, no. 7, pp. 1712–1727, 2019.

[66] S. Sardellitti, S. Barbarossa, and P. Di Lorenzo, "Graph topology inference based on transform learning," in *Proc. IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec 2016, pp. 356–360.

[67] G. U. Yule, "Ii.—a mathematical theory of evolution, based on the conclusions of dr. jc willis, fr s," *Philosophical transactions of the Royal Society of London. Series B, containing papers of a biological character*, vol. 213, no. 402-410, pp. 21–87, 1925.

[68] D. A. Spielman, "Spectral graph theory and its applications," in *48th Annual IEEE Symposium on Foundations of Computer Science (FOCS'07)*. IEEE, 2007, pp. 29–38.

[69] M. Puschel and J. M. Moura, "Algebraic signal processing theory: Foundation and 1-d time," *IEEE Transactions on Signal Processing*, vol. 56, no. 8, pp. 3572–3585, 2008.

[70] R. Varma, S. Chen, and J. Kovačević, "Graph topology recovery for regular and irregular graphs," in *2017 IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Dec 2017, pp. 1–5.

[71] D. Marinakis and G. Dudek, "A practical algorithm for network topology inference," in *Proc. IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.* IEEE, 2006, pp. 3108–3115.

[72] E. Pavez, H. E. Egilmez, and A. Ortega, "Learning graphs with monotone topology properties and multiple connected components," *IEEE Transactions on Signal Processing*, vol. 66, no. 9, pp. 2399–2413, 2018.

[73] J. P. Satya, N. Bhatt, R. Pasumarthy, and A. Rajeswaran, "Identifying topology of power distribution networks based on smart meter data," *arXiv preprint arXiv:1609.02678*, 2016.

[74] E. Pavez and A. Ortega, "Generalized laplacian precision matrix estimation for graph signal processing," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 6350–6354.

[75] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 2826–2830.

[76] M. Azadkia, "Adaptive estimation of noise variance and matrix estimation via usvt algorithm," *arXiv preprint arXiv:1801.10015*, 2018.

[77] P. D. Wentzell, "Measurement errors in multivariate chemical data," *Journal of the Brazilian Chemical Society*, vol. 25, no. 2, pp. 183–196, 2014.

[78] N. P. Galatsanos and A. K. Katsaggelos, "Methods for choosing the regularization parameter and estimating the noise variance in image restoration and their relation," *IEEE Transactions on image processing*, vol. 1, no. 3, pp. 322–336, 1992.

[79] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero, "Learning sparse graphs under smoothness prior," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 6508–6512.

[80] J. P. Keating, R. L. Mason, and P. K. Sen, *Pitman's measure of closeness: a comparison of statistical estimators*. SIAM, 1993.

[81] H. Rue and L. Held, *Gaussian Markov random fields: theory and applications*. CRC press, 2005.

[82] A. T. Basilevsky, *Statistical Factor Analysis and Related Methods: Theory and Applications*. John Wiley & Sons, 2009, vol. 418.

[83] M. Ramezani-Mayiami, M. Hajimirsadeghi, K. Skretting, R. S. Blum, and H. V. Poor, "Graph topology learning and signal recovery via bayesian inference," in *2019 IEEE Data Science Workshop (DSW)*. IEEE, 2019, pp. 52–56.

[84] Z. Bai and G. H. Golub, "Bounds for the trace of the inverse and the determinant of symmetric positive definite matrices," *Annals of Numerical Mathematics*, vol. 4, pp. 29–38, 1996.

[85] J. Löfberg, "YALMIP : A toolbox for modeling and optimization in MATLAB," in *Proc. IEEE International Conference on Robotics and Automation*, Sep. 2004, pp. 284–289.

[86] N. Parikh, S. Boyd *et al.*, "Proximal algorithms," *Foundations and Trends® in Optimization*, vol. 1, no. 3, pp. 127–239, 2014.

[87] H. Bauschke, P. Combettes, and D. Noll, "Joint minimization with alternating bregman proximity operators," *Pacific Journal of Optimization*, 2005.

[88] K. Yosida, *Functional Analysis*. Springer Verlag, 1964.

[89] J.-J. Moreau, "Proximité et dualité dans un espace hilbertien," *Bull. Soc. Math. France*, vol. 93, no. 2, pp. 273–299, 1965.

[90] C. Wang, D. Sun, and K.-C. Toh, "Solving log-determinant optimization problems by a newton-cg primal proximal point algorithm," *SIAM Journal on Optimization*, vol. 20, no. 6, pp. 2994–3013, 2010.

[91] S. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[92] A. W. Roberts and D. E. Varberg, "Another proof that convex functions are locally lipschitz," *The American Mathematical Monthly*, vol. 81, no. 9, pp. 1014–1016, 1974.

[93] C. Hu, L. Cheng, J. Sepulcre, K. A. Johnson, G. E. Fakhri, Y. M. Lu, and Q. Li, "A spectral graph regression model for learning brain connectivity of alzheimer's disease," *PloS one*, vol. 10, no. 5, p. e0128136, 2015.

[94] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.

[95] C. Goutte and E. Gaussier, "A probabilistic interpretation of precision, recall and f-score, with implication for evaluation," in *European Conference on Information Retrieval*. Springer, 2005, pp. 345–359.

[96] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100–103, 2010.

[97] "National climatic data center," ftp://ftp.ncdc.noaa.gov/pub/data/gsod/.

[98] P. J. Rousseeuw and A. M. Leroy, *Robust regression and outlier detection*. John wiley & sons, 2005, vol. 589.

[99] V. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, Oct 2004. [Online]. Available: https://doi.org/10.1023/B:AIRE.0000045502.10941.a9

[100] F. Black, "Noise," *The journal of finance*, vol. 41, no. 3, pp. 528–543, 1986.

[101] C. Luo, Y. Zhao, L. Cao, Y. Ou, and L. Liu, "Outlier mining on multiple time series data in stock market," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2008, pp. 1010–1015.

[102] A. Rahmani, S. Afra, O. Zarour, O. Addam, N. Koochakzadeh, K. Kianmehr, R. Alhajj, and J. Rokne, "Graph-based approach for outlier detection in sequential data and its application on stock market and weather data," *Knowledge-Based Systems*, vol. 61, pp. 89–97, 2014.

[103] K. M. Abadir and J. R. Magnus, *Matrix algebra*. Cambridge University Press, 2005, vol. 1.

[104] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[105] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.1," http://cvxr.com/cvx, Mar. 2014.

[106] P. C. Hansen, "The truncated SVD as a method for regularization," *BIT Numerical Mathematics*, vol. 27, no. 4, pp. 534–553, 1987.

[107] S. Wold, K. Esbensen, and P. Geladi, "Principal component analysis," *Chemometrics and intelligent laboratory systems*, vol. 2, no. 1-3, pp. 37–52, 1987.

[108] Y.-X. Wang and Y.-J. Zhang, "Nonnegative matrix factorization: A comprehensive review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, no. 6, pp. 1336–1353, 2013.

[109] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*. ACM, 2003, pp. 267–273.

[110] D. Cai, X. He, J. Han, and T. S. Huang, "Graph regularized nonnegative matrix factorization for data representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 33, no. 8, pp. 1548–1560, 2011.

[111] N. Shahid, N. Perraudin, V. Kalofolias, G. Puy, and P. Vandergheynst, "Fast robust PCA on graphs," *IEEE Journal of Selected Topics in Signal Processing*, vol. 10, no. 4, pp. 740–756, 2016.

[112] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, "Graph regularized sparse coding for image representation," *IEEE Transactions on Image Processing*, vol. 20, no. 5, pp. 1327–1336, 2011.

[113] J. B. Tenenbaum, V. De Silva, and J. C. Langford, "A global geometric framework for nonlinear dimensionality reduction," *science*, vol. 290, no. 5500, pp. 2319–2323, 2000.

[114] D. Thanou, D. I. Shuman, and P. Frossard, "Parametric dictionary learning for graph signals," in *2013 IEEE Global Conference on Signal and Information Processing*. IEEE, 2013, pp. 487–490.

[115] S. T. Roweis and L. K. Saul, "Nonlinear dimensionality reduction by locally linear embedding," *science*, vol. 290, no. 5500, pp. 2323–2326, 2000.

[116] N. Shahid, V. Kalofolias, X. Bresson, M. Bronstein, and P. Vandergheynst, "Robust principal component analysis on graphs," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 2812–2820.

[117] R. Jenatton, G. Obozinski, and F. Bach, "Structured sparse principal component analysis," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, 2010, pp. 366–373.

[118] D. Thanou, D. I. Shuman, and P. Frossard, "Learning parametric dictionaries for signals on graphs," *IEEE Transactions on Signal Processing*, vol. 62, no. 15, pp. 3849–3862, 2014.

[119] S. Hawe, M. Seibert, and M. Kleinsteuber, "Separable dictionary learning," in *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, 2013, pp. 438–445.

[120] M. Ramezani-Mayiami and K. Skretting, "Topology inference and signal representation using dictionary learning," in *Proc. IEEE 27th European Signal Processing Conference (EUSIPCO)*, Sep. 2019.

[121] R. T. Rockafellar, *Convex analysis*. Princeton university press, 2015.

[122] D. C. Liu and J. Nocedal, "On the limited memory BFGS method for large scale optimization," *Mathematical programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[123] S. Haufe, K.-R. Müller, G. Nolte, and N. Krämer, "Sparse causal discovery in multivariate time series," in *Causality: Objectives and Assessment*, 2010, pp. 97–106.

[124] A. C. Lozano, N. Abe, Y. Liu, and S. Rosset, "Grouped graphical granger modeling for gene expression regulatory networks discovery," *Bioinformatics*, vol. 25, no. 12, pp. i110–i118, 2009.

[125] C. W. Granger, "Some recent development in a concept of causality," *Journal of econometrics*, vol. 39, no. 1-2, pp. 199–211, 1988.

[126] ——, "Investigating causal relations by econometric models and cross-spectral methods," *Econometrica: Journal of the Econometric Society*, pp. 424–438, 1969.

[127] S. Basu, A. Shojaie, and G. Michailidis, "Network granger causality with inherent grouping structure," *The Journal of Machine Learning Research*, vol. 16, no. 1, pp. 417–453, 2015.

[128] A. K. Seth, A. B. Barrett, and L. Barnett, "Granger causality analysis in neuroscience and neuroimaging," *Journal of Neuroscience*, vol. 35, no. 8, pp. 3293–3297, 2015.

[129] A. Lemmens, C. Croux, and M. G. Dekimpe, "Measuring and testing granger causality over the spectrum: An application to european production expectation surveys," *International Journal of Forecasting*, vol. 24, no. 3, pp. 414–431, 2008.

[130] D. Marinazzo, W. Liao, H. Chen, and S. Stramaglia, "Nonlinear connectivity by granger causality," *Neuroimage*, vol. 58, no. 2, pp. 330–338, 2011.

[131] L. Barnett and A. K. Seth, "Behaviour of granger causality under filtering: theoretical invariance and practical application," *Journal of neuroscience methods*, vol. 201, no. 2, pp. 404–419, 2011.

[132] S. L. Bressler and A. K. Seth, "Wiener–granger causality: a well established methodology," *Neuroimage*, vol. 58, no. 2, pp. 323–329, 2011.

[133] Y. Chen, S. L. Bressler, and M. Ding, "Frequency decomposition of conditional granger causality and application to multivariate neural field potential data," *Journal of neuroscience methods*, vol. 150, no. 2, pp. 228–237, 2006.

[134] M. Ding, Y. Chen, and S. L. Bressler, "17 granger causality: basic theory and application to neuroscience," *Handbook of time series analysis: recent theoretical developments and applications*, vol. 437, 2006.

[135] L. Harrison, W. D. Penny, and K. Friston, "Multivariate autoregressive modeling of fmri time series," *Neuroimage*, vol. 19, no. 4, pp. 1477–1491, 2003.

[136] C. W. Anderson, E. A. Stolz, and S. Shamsunder, "Multivariate autoregressive models for classification of spontaneous electroencephalographic signals during mental tasks," *IEEE Transactions on Biomedical Engineering*, vol. 45, no. 3, pp. 277–286, 1998.

[137] K. J. Friston, L. Harrison, and W. Penny, "Dynamic causal modelling," *Neuroimage*, vol. 19, no. 4, pp. 1273–1302, 2003.

[138] M. Ramezani-Mayiami and B. Beferull-Lozano, "Graph recursive least squares filter for topology inference in causal data processes," in *Proc. IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, 2017, pp. 1–5.

[139] D. P. Bertsekas, *Nonlinear programming*. Athena scientific Belmont, 1999.

[140] I. Daubechies, M. Defrise, and C. De Mol, "An iterative thresholding algorithm for linear inverse problems with a sparsity constraint," *Communications on Pure and Applied Mathematics: A Journal Issued by the Courant Institute of Mathematical Sciences*, vol. 57, no. 11, pp. 1413–1457, 2004.

[141] S. J. Wright, R. D. Nowak, and M. A. Figueiredo, "Sparse reconstruction by separable approximation," *IEEE Transactions on Signal Processing*, vol. 57, no. 7, pp. 2479–2493, 2009.

[142] E. T. Hale, W. Yin, and Y. Zhang, "A fixed-point continuation method for l1-regularized minimization with applications to compressed sensing," *CAAM TR07-07, Rice University*, vol. 43, p. 44, 2007.

[143] A. Beck and M. Teboulle, "A fast iterative shrinkage-thresholding algorithm for linear inverse problems," *SIAM journal on imaging sciences*, vol. 2, no. 1, pp. 183–202, 2009.