Learning Sparse Graph Laplacian with K Eigenvector Prior via Iterative GLASSO and Projection

Saghar Bagheri

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF MASTER OF SCIENCE

GRADUATE PROGRAM IN ELECTRICAL ENGINEERING
AND COMPUTER SCIENCE
YORK UNIVERSITY
TORONTO, ONTARIO

JUNE 2021

Abstract

Learning a suitable graph is an important precursor to many graph signal processing (GSP) tasks, such as graph signal compression and denoising. Previous graph learning algorithms either make assumptions on graph connectivity (e.g., graph sparsity), or make individual edge weight assumptions such as positive edges only. In this thesis, given an empirical covariance matrix $\bar{\mathbf{C}}$ computed from data as input, an eigen-structural assumption on the graph Laplacian matrix $\mathbf{L}$ is considered: the first K eigenvectors of $\mathbf{L}$ are pre-selected, e.g., based on domain-specific criteria, and the remaining eigenvectors are then learned from data. One example use case is image coding, where the first eigenvector is pre-chosen to be constant, regardless of available observed data. It is first proven that the subspace $\mathcal{H}_{\mathbf{u}}^{+}$ of symmetric positive semi-definite (PSD) matrices with the first K eigenvectors being $\{\mathbf{u}_k\}_{k=1}^{K}$ in a defined Hilbert space is a convex cone. Then, an operator is constructed which projects a given positive definite (PD) matrix $\mathbf{L}$ to $\mathcal{H}_{\mathbf{u}}^{+}$, inspired by the Gram-Schmidt procedure. Finally, an efficient hybrid graphical lasso / projection algorithm is designed to compute the most suitable graph Laplacian matrix $\mathbf{L}^* \in \mathcal{H}_{\mathbf{u}}^{+}$ given $\bar{\mathbf{C}}$. Experimental results show that given the first K eigenvectors as a prior, this algorithm outperforms competing graph learning schemes using a variety of graph comparison metrics.

Contents

List of Figures

Chapter 1

Introduction

Large modern datasets often exhibit underlying correlation structures. For example, temperature measurements collected by a distribution of wireless sensors in a rain forest are correlated if the sensors are close in physical distance. These correlation structures can be conveniently described by graphs. A graph is composed of nodes and edges. Each node is associated with a datum or measurement (*e.g.*, temperature at a particular sensor), and each edge contains a weight that conveys pairwise relationship between the two connected nodes (*e.g.*, distance between two sensors, or conditional correlation between two random variables). One can find examples of datasets with underlying correlation structures in many real-world settings, such as social networks, wireless sensor networks, transportation networks, and biological networks.

Armed with a graph that accurately describes the underlying correlation structure, computational tools can be designed to process observed data on top of the graph for various signal processing tasks—compression, denoising, interpolation, etc [5, 10]. This is the new and fast-growing field of *graph signal processing* (GSP). Practical GSP applications include image compression [2,11–13], image denoising [14–17], image deblurring [18], point cloud denoising, matrix completion, etc [19–21]. However, often the most suitable graph structure for a dataset is not directly available *a priori* for GSP tools to exploit. As such, an appropriate graph must be first estimated. This thesis focuses on the problem of graph learning from data.

There are three general approaches to learning graphs from data: (i) statistical models, (ii) physically-motivated models and (iii) GSP-based models. From the statistical perspective, the structure of the graph provides a probabilistic description (*e.g.*, joint probability distribution) of the observed data entities, *e.g.*, columns of a data matrix. A classical example is *Gaussian Markov Random Field* (GMRF) [22], where the pairwise correlation information encoded in a graph also implies the probability distribution function. Hence, in the case of GMRF, learning the graph is equivalent to learning a factorization of a joint probability distribution of these random variables. Potential applications are numerous and broad in scope. For example, one can infer interactions between genes using gene expression profiles, or examine the relationships between different politicians given their voting records [23].

Physically-motivated models, however, make assumptions (*e.g.*, sparsity) based on an underlying physical phenomenon or a process on a graph. For example, physically-based methods are used to understand the flow of information within online social networks [24], or to make time-sensitive observations about epidemic spread over a network of people [25].

Finally, GSP-based methods are constructed from the *signal representation* perspective, placing a strong and specific emphasis on the relationship between the graph topology and the representation of signal on the graph. These methods typically assume that the observations are low graph frequency signals, where frequency components are eigenvectors of the adjacency or graph Laplacian matrix [26]. GSP-based methods often make nodal domain assumptions, such as specific graph connectivity or overall sparsity of the graph. These methods can be used in applications that are not based on well-defined physical phenomena or processes. Example applications include image coding and compression, brain signal analysis and political voting analysis [2, 27–29].

None of the above categories of graph learning methods make assumptions about the *eigen-structure* of the graph Laplacian matrix. In this thesis, explicit eigen-structure assumptions on the graph Laplacian matrix $\mathbf{L}$ are introduced into a graph learning framework. Specifically, it is assumed that the first K eigenvectors of $\mathbf{L}$ are pre-selected or computed based on domain-specific criteria. Consider the example of image compression, where one can deduce from domain knowledge that the most common pixel block is the constant signal, and thus should be the first eigenvector regardless of (possibly limited) training data. Consider also political voting records, where the most common pattern is voting along party affiliations in a two-party political system, and thus the first eigenvector should be piecewise constant (*i.e.*, nodes of one political party are assigned 1, while nodes of the other party are -1). There are also practical cases where the first K eigenvectors can be pre-chosen for fast computation. For example, one can use *fast graph Fourier transform* (FGFT) [30] to construct a set of K sparse eigenvectors based on Givens rotation matrices, so that the first K transform coefficients can be computed speedily.

This thesis proposes an optimization strategy where a graph Laplacian matrix $\mathbf{L}$ is learned optimally from data, while restricting the first K eigenvectors to be those chosen ahead of time. It is first proven that the subspace $\mathcal{H}_{\mathbf{u}}^{+}$ of symmetric *positive semi-definite* (PSD) matrices with the first K eigenvectors taken from a given set $\{\mathbf{u}_k\}_{k=1}^K$ is a convex cone. A projection operator is next constructed to project a given positive definite (PD) matrix $\mathbf{L}$ to $\mathcal{H}_{\mathbf{u}}^+$, inspired by the Gram-Schmidt procedure [31]. Finally, an efficient hybrid graphical lasso (GLASSO) / projection algorithm is designed to compute the most suitable graph Laplacian matrix $\mathbf{L}^* \in \mathcal{H}_{\mathbf{u}}^+$ given input empirical covariance matrix $\bar{\mathbf{C}}$. Experiments show that using the first K eigenvector prior, the proposed algorithm can produce a more accurate graph inference given limited observed data compared to existing schemes.

The thesis structure is as follows. Chapter 2 reviews necessary graph signal processing (GSP) and Hilbert space concepts, and different techniques in the graph learning literature. The proposed projection operator is defined in Chapter 3. Chapter 4 describes the proposed graph learning algorithm. Experimental results and conclusion are presented in Chapter 5 and 6, respectively.

Chapter 2

Related Works

This chapter provides an introduction of important concepts in graph signal processing (GSP) in Section 2.1. Definitions needed to define Hilbert space and projection are presented in Section 2.2. In Section 2.3, a brief review of transform coding is provided. In Section 2.4, an overview of different approaches and methods in graph learning is presented.

2.1 Graph Signal Processing Definitions

The field of *Digital Signal Processing* (DSP) studies discrete signals on regular data kernels. For example, DSP contains computational tools to process a digital audio signal on a regularly sampled timeline, or a digital image on a 2D grid. The field of *Graph Signal Processing* (GSP), in contrast, studies signals on irregular data kernels described by graphs [10, 19, 32]. Definitions of graphs and graph signals are first provided in Section 2.1.1. Graph spectrum interpretation and graph construction are then reviewed in Section 2.1.2 and 2.1.3, respectively.

2.1.1 Graph Definitions and Graph Signals

A graph is a mathematical structure encoded with pairwise relations between data entities. A graph consists of nodes, or vertices, which are connected by edges. The edges can be weighted or unweighted. In a weighted graph, a scalar value is assigned to each edge, while in an unweighted graph, all edges have weights equal to 1.

Formally, a graph can be expressed in the form $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$ with a finite set of nodes $\mathcal{V}$ of size $|\mathcal{V}| = N$, a set of edges $\mathcal{E}$, and an *adjacency matrix* $\mathbf{W} \in \mathbb{R}^{N\times N}$. Assuming that graph $\mathcal{G}$ is weighted, an edge $(i, j) \in \mathcal{E}$ connects nodes i and j with weight $w_{ij} \in \mathbb{R}$, where $W_{ij} = w_{ij}$. $W_{ij} = 0$ if $(i, j) \notin \mathcal{E}$. $\mathbf{W}$ may include self-loops W_{ii}'s.

A graph can be directed or undirected. In an undirected graph, an edge (i, j) implies traversal from node i to node j as well as j to i, and $W_{ij} = W_{ji}$. Thus, an undirected graph implies a symmetric $\mathbf{W}$. On the other hand, in a directed graph, edges (i, j) and (j, i) may have different edge weights. $\mathbf{W}$ is not symmetric for an undirected graph in general.

There are different ways to define the weight of an edge connecting nodes i and j. Generally, the definition of an edge between two nodes i and j varies with applications. Edges can represent pairwise similarities, dissimilarities, or distances between distinct feature vectors. Edge weights can also represent statistical quantities such as conditional correlations, which will be discussed in Section 2.4.1.

A graph signal $\mathbf{x}$ on $\mathcal{G}$ is a discrete signal of dimension N, where each signal sample $x_i \in \mathbb{R}$ is assigned to a corresponding node i in $\mathcal{V}$. Thus, a graph signal can be represented as a vector $\mathbf{x} \in \mathbb{R}^N$. The meaning of a graph signal depends on application, such as measurements (*e.g.*, temperature or image pixel intensity) or label information.

2.1.2 Graph Spectrum

Given an adjacency matrix $\mathbf{W}$ for an undirected graph, the diagonal *degree matrix* $\mathbf{D}$ is defined as follows:

$$D_{ii} = \sum_{j} W_{ij}. \tag{2.1}$$

The *combinatorial graph Laplacian matrix* $\mathbf{L}$ [32] is then defined as follows:

$$\mathbf{L} = \mathbf{D} - \mathbf{W}. \tag{2.2}$$

Based on the above definition, $\mathbf{L}$ is symmetric, so it can be eigen-decomposed via the Spectral Theorem as follows [19]:

$$\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\top}, \tag{2.3}$$

where $\boldsymbol{\Lambda}$ is a diagonal matrix with eigenvalues λ_k on the diagonal, and $\mathbf{U}$ is a matrix composed of orthogonal eigenvectors as columns. Assuming that all edge weights are positive in $\mathbf{W}$, it can be proven that $\mathbf{L}$ is *positive semi-definite* (PSD), *i.e.*, eigenvalues of $\mathbf{L}$ are non-negative [19]. Eigenvalues can be interpreted as *graph frequencies*, and their corresponding eigenvectors can be interpreted as *graph Fourier modes*. The set of eigenvectors $\mathbf{U}^{\top}$ collectively define the *graph Fourier transform* (GFT) [10]. Specifically, graph signal $\mathbf{x}$ can be converted to its frequency representation via GFT using the equation $\boldsymbol{\alpha} = \mathbf{U}^\top \mathbf{x}$. GFT can be considered as a generalization of known discrete transforms such as the *discrete cosine transform* (DCT) [33]. Further discussion on DCTs can be found in Section 2.3.2.

There exist different variants of the graph Laplacian matrix, each with distinct properties. $\mathbf{L}_n = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$ is called a *normalized graph Laplacian*, which is a symmetric normalized version of $\mathbf{L}$. $\mathbf{L}_r = \mathbf{D}^{-1}\mathbf{L}$ is a *random walk graph Laplacian*, which is an asymmetric normalized version of $\mathbf{L}$. Finally, to properly account for self-loops, a *generalized graph Laplacian* is defined as $\mathbf{L}_g = \mathbf{L} + \text{diag}(\mathbf{W})$. Each variant also has its own unique spectral properties. For example, $\mathbf{L}_n$ and $\mathbf{L}_r$ have the same eigenvalues $0 \leq \lambda_k \leq 2$. $\mathbf{L}_n$ and $\mathbf{L}$ are both symmetric, but $\mathbf{L}_n$ does not have the constant vector as a first eigenvector.

2.1.3 Graph Construction

Different applications of GSP use different methods to define pairwise similarities (edge weights) between nodes to construct a graph [9, 34, 35]. In a temperature sensing system, for example, each sensor has a known physical 2D location [34]. In this case, pairwise similarities between nodes of a graph can be defined in terms of physical distances in space between pairs of sensors. In machine learning applications, however, each data entity, represented by a graph node i, may be endowed with a feature vector $\mathbf{f}_i \in \mathbb{R}^K$ of dimension K. In this instance, pairwise similarities are defined given a notion of distance between vectors $\mathbf{f}_i$ in feature space. More generally, an edge weight w_{ij} between nodes i and j in a graph can be computed based on the distance between two nodes, d_{ij}, and a Gaussian kernel with parameter σ:

$$w_{ij} = \exp\left(-\frac{d_{ij}^2}{2\sigma^2}\right). \tag{2.4}$$

The *bilateral filter* [36, 37] is one distance notion commonly used in image processing. It is defined as follows:

$$w_{ij} = \exp\left(-\frac{\|\mathbf{l}_i - \mathbf{l}_j\|_2^2}{2\sigma_l^2}\right) \exp\left(-\frac{|x_i - x_j|_2^2}{2\sigma_x^2}\right), \tag{2.5}$$

where σ_l and σ_x are parameters, and $\mathbf{l}_i$ and x_i are the respective location and pixel intensity of pixel (node) i on the image grid. Some works in image processing [16] also use the simpler form of (2.5), where only pixel intensities are considered to compute edge weights:

$$w_{ij} = \exp\left(-\frac{|x_i - x_j|^2}{2\sigma_x^2}\right). \tag{2.6}$$

(2.6) means that an edge weight is small when two pixels have a large difference in intensity, such as pixels on two different sides of an object boundary.

In [8, 27], a statistical notion of edge weight is used, which will be discussed in greater detail in Section 2.4.1. Special attention will be paid to *Gaussian Markov Random Fields (GMRFs)* [38, 39]. Assuming a GMRF probabilistic model, the precision matrix can be learned such that it has the form of a graph Laplacian [27].

2.2 Hilbert Spaces

The following terminologies from [5] are used to define a Hilbert space. These definitions are necessary to construct a projection operator in Chapter 3.

The definition of a vector space is presented in Section 2.2.1. The definitions for inner product and orthogonality (Section 2.2.2), norm (Section 2.2.3), Hilbert space (Section 2.2.4), and linear and adjoint operators (Section 2.2.5) are discussed next. Finally, a projection operator is defined in Section 2.2.6.

2.2.1 Vector Space

A vector space, over a field of scalars $\mathbb{R}$, is a set of vectors $\mathcal{V}$ together with operations of scalar multiplication and vector addition. A vector space has the following properties: (Given $\mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{V}$ and scalars $\alpha, \beta \in \mathbb{R}$)

$$\begin{aligned}
&\mathbf{x}+\mathbf{y}=\mathbf{y}+\mathbf{x} && \text{(Commutativity)}\\
&(\mathbf{x}+\mathbf{y})+\mathbf{z}=\mathbf{x}+(\mathbf{y}+\mathbf{z}) \;\text{ and }\; (\alpha\beta)\mathbf{x}=\alpha(\beta\mathbf{x}) && \text{(Associativity)}\\
&\alpha(\mathbf{x}+\mathbf{y})=\alpha\mathbf{x}+\alpha\mathbf{y} \;\text{ and }\; (\alpha+\beta)\mathbf{x}=\alpha\mathbf{x}+\beta\mathbf{x} && \text{(Distributivity)}
\end{aligned} \tag{2.7}$$

Furthermore, a vector space has the following properties:

$$\begin{aligned}
&\exists\mathbf{0}\in\mathcal{V} \;\text{ s.t. }\; \mathbf{x}+\mathbf{0}=\mathbf{x},\;\; \forall\mathbf{x}\in\mathcal{V} && \text{(Additive identity)}\\
&\exists-\mathbf{x}\in\mathcal{V} \;\text{ s.t. }\; \mathbf{x}+(-\mathbf{x})=\mathbf{0},\;\; \forall\mathbf{x}\in\mathcal{V} && \text{(Additive inverse)}\\
&1\cdot\mathbf{x}=\mathbf{x},\;\; \forall\mathbf{x}\in\mathcal{V} && \text{(Multiplicative identity)}
\end{aligned} \tag{2.8}$$

A nonempty subset $\mathcal{S}$ of a vector space $\mathcal{V}$ is a *subspace* when it is closed under the operations of scalar multiplication and vector addition:

$$\forall \mathbf{x}, \mathbf{y} \in \mathcal{S},\ \mathbf{x} + \mathbf{y} \in \mathcal{S} \tag{2.9}$$

$$\forall \mathbf{x} \in \mathcal{S} \text{ and } \alpha \in \mathbb{R},\ \alpha\mathbf{x} \in \mathcal{S} \tag{2.10}$$

2.2.2 Inner Product and Orthogonality

An inner product is a function with real values defined on a vector space $\mathcal{V}$ with the following properties: ($\forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathcal{V}$ and $\alpha \in \mathbb{R}$)

$$\begin{aligned}
&\langle \mathbf{x} + \mathbf{y}, \mathbf{z} \rangle = \langle \mathbf{x}, \mathbf{z} \rangle + \langle \mathbf{y}, \mathbf{z} \rangle && \text{(Distributivity)}\\
&\langle \alpha\mathbf{x}, \mathbf{y} \rangle = \alpha \langle \mathbf{x}, \mathbf{y} \rangle && \text{(Linearity in the first argument)}\\
&\langle \mathbf{x}, \mathbf{y} \rangle^\star = \langle \mathbf{y}, \mathbf{x} \rangle && \text{(Hermitian symmetry)}\\
&\langle \mathbf{x}, \mathbf{x} \rangle \geq 0, \quad \text{and} \quad \langle \mathbf{x}, \mathbf{x} \rangle = 0 \ \text{ iff } \ \mathbf{x} = \mathbf{0} && \text{(Positive definiteness)}
\end{aligned} \tag{2.11}$$

An inner product can endow a space with some geometric properties such as *orthogonality*. Vectors $\mathbf{x}$ and $\mathbf{y}$ are said to be *orthogonal* (*i.e.*, $\mathbf{x} \perp \mathbf{y}$) when $\langle \mathbf{x}, \mathbf{y} \rangle = 0$.

2.2.3 Norm

A norm is a function which allocates size (*e.g.*, length) to vectors. More specifically, a norm is a real-valued function defined on a vector space $\mathcal{V}$ over $\mathbb{R}$ with the following properties: ($\forall \mathbf{x}, \mathbf{y} \in \mathcal{V}$ and $\alpha \in \mathbb{R}$)

$$\begin{aligned}
&\|\mathbf{x}\| \geq 0, \quad \text{and} \quad \|\mathbf{x}\| = 0 \ \text{ iff } \ \mathbf{x} = \mathbf{0} && \text{(Positive definiteness)}\\
&\|\alpha\mathbf{x}\| = |\alpha| \|\mathbf{x}\| && \text{(Positive scalability)}\\
&\|\mathbf{x} + \mathbf{y}\| \leq \|\mathbf{x}\| + \|\mathbf{y}\| && \text{(Triangle inequality)}
\end{aligned} \tag{2.12}$$

When a norm operation exists in a vector space, the vector space is a normed vector space. In a normed vector space, the metric, or distance, between two vectors $\mathbf{x}$ and $\mathbf{y}$ is the norm (*e.g.*, the size) of their difference:

$$\mathbf{d}(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|. \tag{2.13}$$

A subspace $\mathcal{S}$ of a normed vector space $\mathcal{V}$ is said to be *closed* when it contains all limits of sequences of vectors in $\mathcal{S}$. Subspaces of normed vector spaces with finite dimensions are always closed.

2.2.4 Hilbert Space

Formally, a sequence of vectors $\mathbf{x}_1, \ldots$ in a normed vector space $\mathcal{V}$ *converges* to $\mathbf{v} \in \mathcal{V}$ when $\lim_{k\to\infty} \|\mathbf{v} - \mathbf{x}_k\| = 0$. Specifically, given any $\epsilon > 0$, there exists a K_ϵ such that:

$$\|\mathbf{v} - \mathbf{x}_k\| < \epsilon, \quad \forall k > K_\epsilon. \tag{2.14}$$

Given (2.14), the meaning of convergence depends on the definition of the norm in each subspace.

A sequence of vectors $\mathbf{x}_1, \ldots$ in a normed vector space is called a *Cauchy sequence* when, given any $\epsilon > 0$, there exists a K_ϵ such that

$$\|\mathbf{x}_k - \mathbf{x}_m\| < \epsilon \quad , \forall k, m > K_\epsilon. \tag{2.15}$$

For sequences with real value, a Cauchy sequence must converge.

Completeness means convergence exists in a subspace. More formally, a *complete* space is a normed vector space $\mathcal{V}$ in which every Cauchy sequence in $\mathcal{V}$ converges to a vector in $\mathcal{V}$. A *Hilbert space* is an inner product space with the additional requirement of completeness. Figure 2.1 shows the relationships among different types of spaces and provides examples of standard spaces.

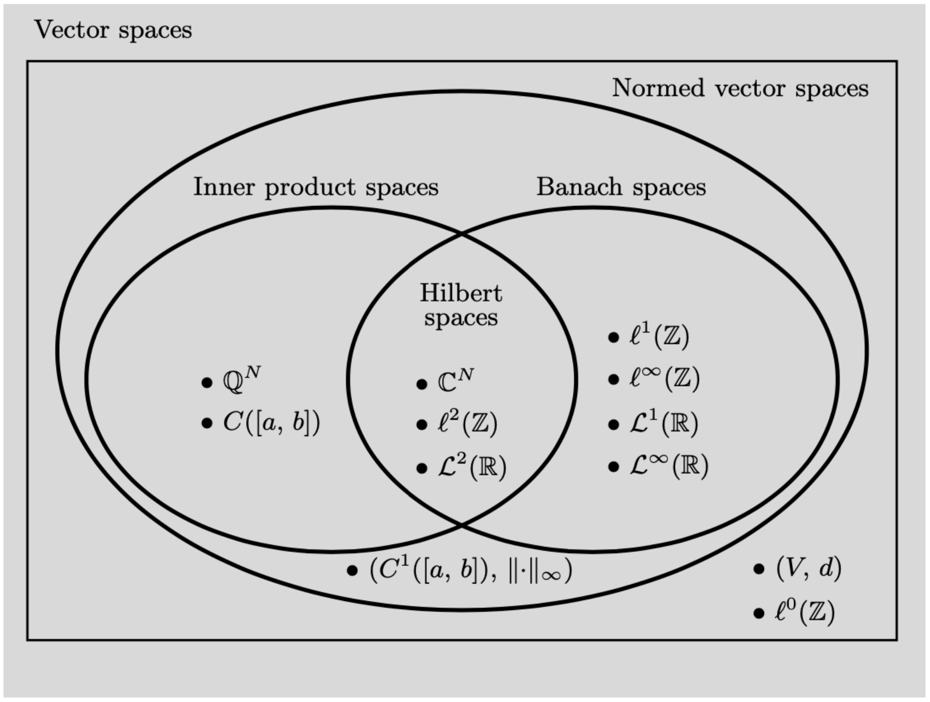


Figure 2.1: Different types of vector spaces [1].

2.2.5 Linear Operator and Adjoint Operator

A function $\mathbf{A} : \mathcal{H}_0 \to \mathcal{H}_1$ is called a *linear operator* from Hilbert space $\mathcal{H}_0$ to Hilbert space $\mathcal{H}_1$ when, for all $\mathbf{x}, \mathbf{y}$ in $\mathcal{H}_0$ and α in $\mathbb{R}$, the following properties hold:

$$\begin{aligned} \mathbf{A}(\mathbf{x} + \mathbf{y}) &= \mathbf{A}\mathbf{x} + \mathbf{A}\mathbf{y} \quad &\text{(Additivity)} \\ \mathbf{A}(\alpha\mathbf{x}) &= \alpha(\mathbf{A}\mathbf{x}) \quad &\text{(Scalability)} \end{aligned} \tag{2.16}$$

When the domain $\mathcal{H}_0$ and the co-domain $\mathcal{H}_1$ are the same, $\mathbf{A}$ is also called a linear operator on $\mathcal{H}_0$.

The linear operator $\mathbf{A}^* : \mathcal{H}_1 \to \mathcal{H}_0$ is called the *adjoint* of the linear operator $\mathbf{A} : \mathcal{H}_0 \to \mathcal{H}_1$ with the following property:

$$\langle \mathbf{A}\mathbf{x}, \mathbf{y} \rangle_{\mathcal{H}_1} = \langle \mathbf{x}, \mathbf{A}^*\mathbf{y} \rangle_{\mathcal{H}_0}, \ \forall \mathbf{x} \in \mathcal{H}_0, \forall \mathbf{y} \in \mathcal{H}_1. \tag{2.17}$$

When $\mathbf{A} = \mathbf{A}^*$, the operator $\mathbf{A}$ is said to be *self-adjoint* or *Hermitian.*

2.2.6 Projection Operator

An *idempotent* operator $\mathbf{P}$ is an operator such that:

$$\mathbf{P}^2 = \mathbf{P}. \tag{2.18}$$

A *projection* operator is a bounded linear operator that is idempotent. An *orthogonal projection* operator is a projection operator that is self-adjoint. An *oblique projection* operator is a projection operator that is not self-adjoint.

2.3 Transform Coding

Transform coding [40] is one of the most popular approaches for image and video compression. Many compression codecs such as JPEG, HEVC and H.26x [41–43] divide an image into non-overlapping blocks of pixels (*e.g.*, 8×8 pixels) and then transform-encode individual blocks in sequence. This so-called block-based transform coding projects each block using a chosen transform. Different transforms are used in image and video coding literature [11,44]. The *Discrete Cosine Transform* (DCT) and the *Asymmetric Discrete Sine Transform* (ADST) [33, 45] are two of the transforms that are widely used in image compression systems.

Section 2.3.1 provides a review of some related works in transform coding and image coding. The two most popular transforms in image and signal processing—DCT and ADST—are described in Section 2.3.2 and 2.3.3, respectively.

2.3.1 Image Coding

In transform coding [40], an image is divided into non-overlapping blocks of pixels, then a chosen transform is applied to each block. Next, the computed transform coefficients are quantized and entropy-coded into a bitstream for storage or transmission. At the decoder, entropy decoding, inverse quantization and inverse transform are performed to reconstruct the block. To achieve the best compression performance, selecting a transform that promotes sparsity in signal representation is necessary. It means that there are few non-zero transform coefficients remaining following signal transformation and quantization.

JPEG [41] was the first widely adopted image coding standard. The first step of JPEG compression is the division of an image into non-overlapping 8×8 blocks. Next, an individual DCT transform [33] is applied on every pixel block before employing quantization and entropy coding using Huffman code [46]. Huffman code is a simple variable-length-coding (VLC) algorithm used to encode sequences of symbols with different probabilities into a binary string via a Huffman tree.

Contemporary compression methods, such as a coding algorithm in [11] for piecewise smooth images like depth maps, utilize more general GFTs for transform coding of images. Recall that a GFT is effectively a matrix containing eigenvectors of a graph Laplacian matrix. The algorithm in [11] specifies a GFT by designating an underlying graph. This is possible through parameterization of either a weighted or an unweighted graph. The key point in this work is that the compression cost of coding quantized GFT coefficients are weighted against the overhead of describing a graph for correct GFT decoding at the decoder. It is also shown that the derived GFT approximates the ideal *Karhunen-Loève Transform* (KLT) [47]. KLT is composed of eigenvectors of the covariance matrix of the input process (*i.e.*, pixel data), and it is similar to *principal component analysis* (PCA) [48]. Unlike common transforms such as DCT [33], KLT is a signal-adaptive transform. Signal adaptivity allows the derived transform to match against the time-varying statistics of the input signal. However, a disadvantage of KLT is that the transform matrix must be known to both the encoder and decoder. Thus, using KLT can slow down the compression process. To compensate for this, KLT should be approximated to allow for fast computation.

A revised algorithm [2] improved upon the performance of [11] by using *generalized GFT* (GGFT) optimized for intra-prediction residues. GGFT is defined based on the orthogonal eigenvectors of the generalized graph Laplacian matrix. A novel clustering method is also proposed to identify different correlation types (*i.e.*, strong correlation, weak correlation and zero correlation) between pixel pairs. Figure 2.2 illustrates subjective quality comparison among three compression methods using DCT, GFT and GGFT in their coding scheme.

2.3.2 Discrete Cosine Transform

Discrete cosine transform (DCT) [33] is the most commonly used transform for image and video coding in the literature [41–43]. DCT is derived from standard Laplacian

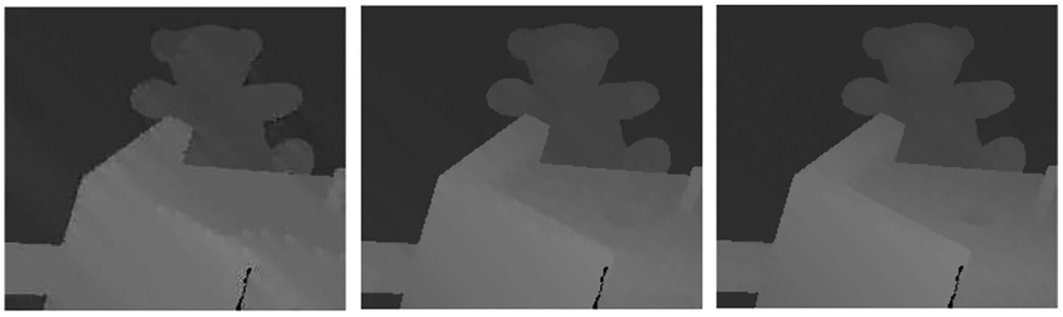

(a) (b) (c)

Figure 2.2: Compression result via **(a)** DCT **(b)** GFT **(b)** GGFT [2].

matrices in discrete domain, and hence it can be interpreted as a GFT, discussed in Section 2.1.2. Specifically, DCT is the GFT for a line graph with equal edge weights. A line graph with N nodes can represent N pixels along a pixel row. Edge weights between all adjacent pixels are 1. The graph signal that corresponds to this graph would be the image intensity at each of those pixels. The combinatorial graph Laplacian for this graph when $N = 8$ is

$$\mathbf{L} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}. \tag{2.19}$$

GFT of this Laplacian is called *DCT-2*. There are four different types of DCTs, which correspond to different Laplacian matrices. The Laplacian matrix for *DCT-1* is

$$\mathbf{L} = \begin{pmatrix} 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -2 & 2 \end{pmatrix}. \tag{2.20}$$

The Laplacian matrix for *DCT-3* is

$$\mathbf{L} = \begin{pmatrix} 2 & -2 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 2 \end{pmatrix}. \tag{2.21}$$

The Laplacian matrix for *DCT-4* is

$$\mathbf{L} = \begin{pmatrix} 1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 3 \end{pmatrix}. \tag{2.22}$$

As shown in these matrices, the differences among the four types of DCTs are due to different boundary conditions [33]. DCT-2 and DCT-4 are mostly used in image processing tasks. All the mentioned types of DCTs are 1D DCTs, and they can be extended to two-dimensional DCTs, which is discussed next.

Formally, if $\mathbf{X}$ is an $N \times N$ block of pixels and $\mathbf{U}_{\mathrm{DCT}}$ is the DCT matrix, then the block after applying transform is as follows [3]:

$$\mathbf{Y} = \mathbf{U}_{\mathrm{DCT}}^{\top}\mathbf{X}\mathbf{U}_{\mathrm{DCT}} \tag{2.23}$$

where $\mathbf{U}_{\mathrm{DCT}}^{\top}$ applies the DCT to the columns of $\mathbf{X}$, while $\mathbf{U}_{\mathrm{DCT}}$ does the same thing for the rows of $\mathbf{X}$. In other words, 2D DCT is a *separable transform*, separable into 1D x- and y-directions.

2.3.3 Asymmetric Discrete Sine Transform

The asymmetric discrete sine transform (ADST) is motivated by intra-prediction and residual encoding [45]. First, images are divided into blocks without overlaps. In classical intra-prediction, decoded pixels from a neighboring block *i.e.*, boundary pixels, are used to predict pixels in the present block *i.e.*, predicted pixels [2]. This basically changes the resulting residual block statistics in the encoding process. For example, after intra-prediction, the constant signal is not the most likely signal anymore for the residues, since the pixel at the prediction boundary tends to be closer to zero than the others. [45] showed that ADST outperforms DCT for these kind of residual blocks. ADST is the "generalized" GFT of a line graph with the following "generalized" graph Laplacian:

$$\mathbf{L} = \begin{pmatrix} 2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \end{pmatrix}. \tag{2.24}$$

The only difference between ADST and DCT in (2.19) is changing 2 to 1 in entry $(1, 1)$ of the Laplacian matrix. This is equal to adding a *self-loop* to the first and left-most node in the line graph. This results in changing the eigenvector corresponding to the lowest frequency, so that it is no longer a constant vector. Specifically, the difference between ADST and DCT can be illustrated by comparing their basis vectors in Figure 2.3. As shown, for ADST, the value for the left-most node becomes smaller, particularly for the first basis.

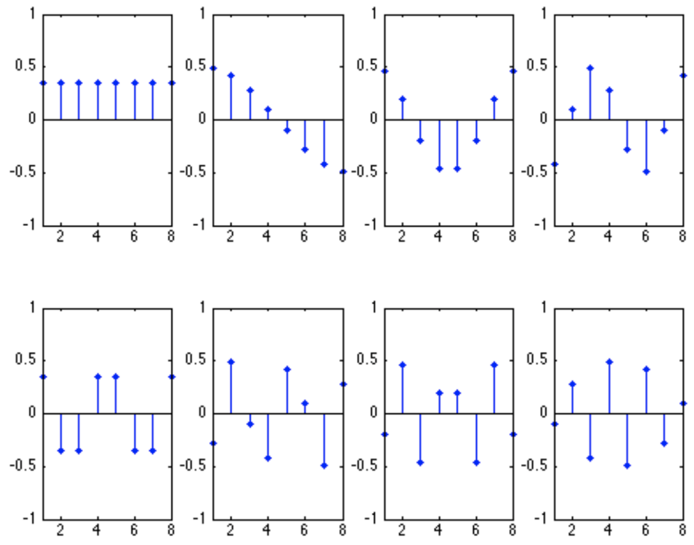
2.4 Graph Learning Techniques

As discussed in Chapter 1, previous graph learning methods can be divided into three main categories. In this chapter, first, previous approaches that address the graph learning problem from the statistical or physical perspective are overviewed in Section 2.4.1 and 2.4.2, respectively. Next, recent GSP-based approaches are reviewed in Section 2.4.3, showing how additional constraints can be deduced from the GSP perspective, so that an appropriate graph can be learned robustly, even with a small dataset.

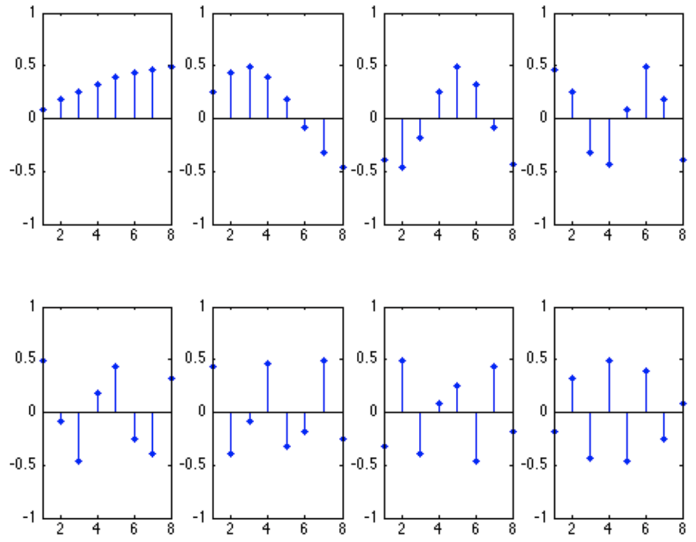
2.4.1 Statistical Models

From the statistical perspective, let the columns of the data matrix $\mathcal{X}$ be data observations generated from a same probability model. Using $\mathcal{X}$, one can estimate parameters of a statistical model, represented by a graph $\mathcal{G}$. Such models are known as probabilistic graphical models [8,22,23,49,50], where the edges in the graph encode conditional dependencies between the random variables on the nodes of the graph.

These models have two main types. First one is based on undirected graphical models, also known as *Markov random fields* (MRFs). In this section, approaches for learning MRFs are overviewed. These approaches compute a most probable probability model given observations. They are also further extended in GSP-based methods.



(a)



(b)

Figure 2.3: (a) Discrete cosine transform (DCT) basis for vectors of dimension 8. **(b)** Asymmetric Discrete sine transform (ADST) basis for vectors of dimension 8. These vectors are ordered (left to right, top to bottom) by their frequency [3].

MRF with respect to a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the node set and $\mathcal{E}$ denotes the edge set, is a set of discrete random variables $\mathbf{x} = \{x_i : i \in \mathcal{V}\}$ that satisfy the Markov property. Specifically, the pairwise Markov property is defined as follows:

$$(i,j) \notin \mathcal{E} \iff p(x_i|x_j, \mathbf{x} \setminus \{x_i, x_j\}) = p(x_i|\mathbf{x} \setminus \{x_i, x_j\}). \tag{2.25}$$

(2.25) states that two variables x_i and x_j are *conditionally independent* given the knowledge of the rest of the nodes, if there exists no edge between their corresponding nodes i and j in the graph $\mathcal{G}$.

One frequently used example of MRFs is *Gaussian graphical models* or *Gaussian MRFs* (GMRFs) [38, 39]. The joint probability of a zero-mean GMRF for random vector $\mathbf{x} \in \mathbb{R}^N$ is defined as follows:

$$p(\mathbf{x}|\boldsymbol{\Theta}) = \frac{|\boldsymbol{\Theta}|^{1/2}}{(2\pi)^{N/2}} \exp\left(-\frac{1}{2}\mathbf{x}^\top \boldsymbol{\Theta} \mathbf{x}\right) \tag{2.26}$$

where $\boldsymbol{\Theta}$ is the *precision* or inverse covariance matrix. Learning the precision matrix $\boldsymbol{\Theta}$, which contains pairwise correlation of variables, is equivalent to learning the underlying graph structure.

The second type of graphical models are based on directed models, also known as *Bayesian networks* or *belief networks* (BNs). The focus of this thesis is restricted only to learning of undirected graphs. More detailed comparison between MRFs and BNs is provided in [22, 51].

There are different ways to learn GMRFs in the literature. In [52], the authors proposed to learn a covariance matrix by introducing zero entries in the inverse of the sample covariance matrix. This method basically learns a covariance matrix by thresholding small values in the *inverse* covariance matrix. However, it is not applicable when the sample size is smaller than the number of variables, in which case the sample covariance matrix is not invertible.

In [49], the authors introduced a regression method using Lasso [53]. Specifically, this method individually approximates each variable as a sparse linear combination of the observations of other variables given n i.i.d. observations $\mathbf{X} = (\mathbf{X}_1, ..., \mathbf{X}_n)$. This method can be interpreted as a standard regression problem with ℓ_1 regularization and can be solved via Lasso individually for each node in the graph.

For example, to estimate the first variable x_1, a Lasso regression problem [53] can be formulated as:

$$\min_{\boldsymbol{\beta}_1} \|\mathbf{X}_1 - \mathbf{X}_{\backslash 1}\boldsymbol{\beta}_1\|_2^2 + \lambda\|\boldsymbol{\beta}_1\|_1 \tag{2.27}$$

where $\mathbf{X}_1$ (*i.e.*, a transpose of the first row of $\mathbf{X}$) is the observations on variable x_1 and $\mathbf{X}_{\backslash 1}$ is the observations on variables except x_1, λ is a regularization parameter, and $\boldsymbol{\beta}_1 \in \mathbb{R}^{N-1}$ is a vector of coefficients for x_1. In (2.27), the first term is the negative log likelihood, and the second term, *i.e.*, the ℓ_1 norm, is a prior used to enforce sparsity of $\boldsymbol{\beta}_1$. A connection between a pair of nodes v_i and v_j in a graph exists if either β_{ij}, β_{ji} or both are nonzero. Note that β_{ij} and β_{ji} are not directly related to the corresponding entries in the precision matrix.

The estimation of a sparse inverse covariance matrix was then studied in several works [49, 54, 55]. In one seminal work, Friedman et al. formulated the sparse inverse covariance estimation problem in a regularization framework, and developed the popular *Graphical Lasso* algorithm for the optimization problem [8]. Graphical Lasso (GLASSO) aims to solve the following problem:

$$\max_{\boldsymbol{\Theta}} \quad \log\det\boldsymbol{\Theta} - \mathrm{Tr}(\boldsymbol{\Sigma}\boldsymbol{\Theta}) - \rho\,\|\boldsymbol{\Theta}\|_1 \tag{2.28}$$

where $\mathbf{\Theta}$ is the sought precision matrix, $\mathbf{\Sigma}$ is the input sample covariance matrix, and $\rho > 0$ is a weight parameter for the sparsity term. The first two terms together can be interpreted as the log likelihood under a GMRF. The entry-wise ℓ_1-norm of $\mathbf{\Theta}$ is added to promote sparsity of graph connections. The major difference between the method in [49] and GLASSO is that the optimization in the first method is independent for each node, while in GLASSO, joint optimization is used, which is crucial for robustness against noise.

In [56], the precision matrix is further constrained to be in the family of *M-matrices* [57]. M-matrices are positive definite and symmetric matrices with non-positive off-diagonal entries. Graph Laplacian matrix for a positive graph is one example of M-matrices. These M-matrices result in *attractive GMRFs* which are defined in [56]. Specifically, if the precision matrix is an M-matrix, then all the pairwise correlations are non-negative, and the corresponding GMRF is called attractive.

One example of learning graph Laplacian is [58]. The authors used the formulation in (2.28) in addition to a constraint that forces the precision matrix to be a Laplacian matrix as follows:

$$\max_{\mathbf{\Theta},\sigma^2} \quad \log\det\mathbf{\Theta} - \mathrm{Tr}(\frac{1}{m}\mathbf{X}\mathbf{X}^\top\mathbf{\Theta}) - \rho\,\|\mathbf{\Theta}\|_1, \tag{2.29}$$

$$\text{s.t} \quad \mathbf{\Theta} = \mathbf{L} + \frac{1}{\sigma^2}\mathbf{I}, \quad \mathbf{L} \in \mathcal{L} \tag{2.30}$$

where m is the number of observations, $\mathcal{L}$ is a set of eligible Laplacian matrices that are PSD and symmetric, $\mathbf{I}$ is the identity matrix, and $\sigma^2 > 0$ is the variance parameter that can be interpreted as the *a priori* feature variance [59]. Further extensions of GLASSO are presented in [23, 60], and some computationally efficient methods are discussed in [61, 62].

2.4.2 Physically-motivated Models

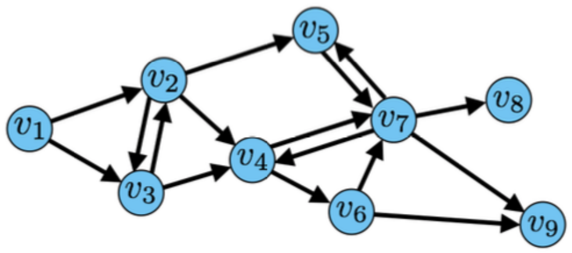
These models address the graph learning problem by taking a physically-motivated approach. In such cases, the observations are interpreted as the results of some physical phenomenon on the graph, and the graph learning problem includes capturing natural structure from the physics of the observed data.

One example is in the network tomography field, which generally uses physically-based assumptions to estimate network structures from implicit observations [63]. Network tomography is most often used in the telecommunication networks, where the expected inferred information may contain network routes or features such as available bandwidth or reliability of each link in the network. For example, in [64] the graph structure was modeled as a tree, and it was assumed that information packets are sending from one source to multiple destinations. In contrast, the topology used in [65] was based on the assumption that information packets were sent from multiple sources to the same destination.

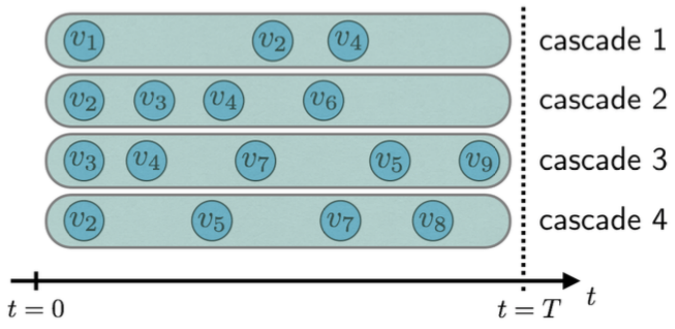
Another example is information propagation models that have been implemented to infer underlying biological, social, and financial networks based on observations of epidemics, or other signal diffusion over them [4, 24, 66, 67]. Given a known graph structure, epidemic processes over graphs have been well-studied, where nodes are vulnerable and can be infected (and subsequently healed) [68]. However, when the structure is unknown in advance, a graph can be estimated from the evolution of a contagion over the graph nodes.

One of the popular physical models to represent propagation over time is *cascade.*

Formally, a fully-observed cascade can be defined by the sequence of $\{(v'_p, v_p, t_p)\}_{p=0}^{P}$ where $P \leq N$. This sequence means that node v'_p infected its neighbor v_p at time t_p. Figure 2.4 [4] illustrates a possible direction of diffusion in the graph and observations of cascade over the graph.



(a)



(b)

Figure 2.4: (a) Directed graph showing possible directions of propagation.
(b) Observations of cascades spreading over the graph [4, 5].

Graph learning methods that use information cascades to compute solutions are classifying into two different categories: *homogeneous* or *heterogeneous* models. Homogeneous methods assume cascades propagate in the same way statistically on all edges. For example, in [66], it is assumed that element w_{ij} of the adjacency matrix is equal to conditional probability that node i infects j, if i is already infected.

Heterogeneous methods have more relaxed requirements and assume that cascades can propagate with different rates over different edges. In [4], this type of method is used in an algorithm called *NETRATE*. NETRATE is an efficient inference algorithm that uses stochastic convex optimization to compute online estimates of the edges and transmission rates.

2.4.3 GSP-based models

In this group of techniques, the graph learning problem is posed assuming that observed signals fit some pre-defined graph frequency properties. As discussed in Section 2.1, graph frequencies are typically derived from eigen-structure of the adjacency or graph Laplacian matrix that fully characterizes the combinatorial graph [26]. GSP-based graph learning is popularly used for many recent graph spectral signal restoration schemes, including image denoising, dequantization, deblurring, and contrast enhancement [18, 69–72].

A digital image naturally resides on a regular 2D grid, but can be interpreted as a signal on a graph, where each pixel is represented by a graph node, and an edge reflects inter-pixel similarity between two adjacent pixels. The crux in graph construction for an image thus lies in the estimation of these inter-pixel similarities to establish edge weights, given only corrupted and/or partial image observations. Beyond image processing applications, graph learning approaches in GSP literature have been applied to a wide range of applications [73–77].

Some applications in GSP require directed graphs to more suitably describe the datasets. In this type of graphs, the directions of edges may be interpreted as causal relationships between variables that the nodes represent. For instance, in brain signals, graph connections can convey additional information about the causal dependencies between brain regions [28]. GSP-based graph learning approaches have also been applied to derive graphs that represent urban traffic flows, patterns of news spread on social media [78], inter-region political relationship, and similarities among animal species from evolutionary aspects [6]. The variety of these areas has proven the capability of applying GSP-based graph learning methods to discover hidden relationships behind data observations in real world applications.

In this section, two classes of graph learning models in GSP are discussed in detail. First, in Section 2.4.3.1, an overview of recent papers in GSP that use the *Graph Laplacian Regularizer* (GLR) in their graph learning framework is provided. Next, in Section 2.4.3.2, papers that use the same optimization framework as GLASSO are introduced [6, 7].

2.4.3.1 Models based on GLR

Smoothness of signal $\mathbf{x}$ on a graph $\mathcal{G}$ is commonly defined using the *Graph Laplacian Regularizer* (GLR) [15, 16, 20, 70, 79]. One way to learn a graph (or equivalently its Laplacian matrix $\mathbf{L}$), is to minimize the signal variation or GLR on the graph, *i.e.*, to minimize $\mathbf{x}^\top \mathbf{L}\mathbf{x}$. GLR is defined as follows:

$$\mathbf{x}^\top \mathbf{L}\mathbf{x} = \sum_{(i,j)\in\mathcal{E}} w_{i,j}(x_j - x_i)^2 = \sum_{k=1}^{N} \lambda_k {\alpha_k}^2 \tag{2.31}$$

where λ_k's are the eigenvalues of $\mathbf{L}$, and α_k's are the GFT coefficients of signal $\mathbf{x}$. A signal $\mathbf{x}$ is smooth if its GLR is small [16, 58]. In the nodal domain, GLR of signal $\mathbf{x}$ is small when the nodes connected by large edge weights have similar signal intensities. In the graph frequency domain, GLR of $\mathbf{x}$ is small when the energy of high graph frequencies is small. Note that GLR is a measure for global signal smoothness on $\mathcal{G}$. Thus, a small GLR means a small signal variation over all edges in the graph, and the energy of the signal is concentrated in the low-frequency components in the graph spectral domain. Consequently, minimizing (2.31) means low-pass filtering.

The approach of minimizing GLR or its variants with powers of $\mathbf{L}$ has been studied in several existing GSP articles, such as [58, 80]. In another example, in [75, 81] the authors proposed a method to learn a graph for weather stations using the same GLR model.

In [9], the following optimization problem is proposed, inspired by [82]:

$$\min_{\mathbf{L},\mathbf{Y}} \|\mathbf{X}-\mathbf{Y}\|_F^2 + \alpha \mathrm{Tr}(\mathbf{Y}^\top \mathbf{L}\mathbf{Y}) + \beta\|\mathbf{L}\|_F^2, \qquad \text{s.t. } \mathrm{Tr}(\mathbf{L}) = N, \quad \mathbf{L} \in \mathcal{L} \tag{2.32}$$

where $\mathbf{X}$ is a data matrix, and $\|.\|_F$ is the *Frobenius* norm of a matrix. α and β are regularization parameters, and $\mathcal{L}$ is a set of valid Laplacian matrices which are PSD and symmetric. (2.32) seeks a solution $\mathbf{Y}$ that is close to data matrix $\mathbf{X}$, while ensures that $\mathbf{Y}$ is smooth on the graph represented by its Laplacian matrix $\mathbf{L}$. The trace constraint acts as a normalization factor, and the Frobenius norm of $\mathbf{L}$ helps control the distribution of the edge weights *i.e.*, the uniformity of the connection weights, which is motivated by [83].

2.4.3.2 Models based on GLASSO

In this class of models, the proposed algorithms use the same objective function as GLASSO, but with additional constraints to ensure the obtained inverse covariance matrix has specific properties [6,7]. For example, in [6], the authors used additional constraints to guarantee that an inverse covariance matrix in a solution is also a generalized graph Laplacian (or diagonally dominant generalized graph Laplacian or combinatorial graph Laplacian) matrix corresponding to a positive graph. The optimization problem to estimate a generalized graph Laplacian (GGL) matrix is formulated as follows:

$$\min_{\mathbf{\Theta}} \ \operatorname{Tr}(\mathbf{\Theta K}) - \log\det\mathbf{\Theta} \ \text{ subject to } \ \mathbf{\Theta} \in \mathcal{L}_g(\mathbf{A}) \tag{2.33}$$

where $\mathbf{K} = \mathbf{S} + \mathbf{H}$, $\mathbf{H}$ is the regularization matrix, $\mathbf{S}$ is the data statistic, $\mathbf{A}$ is the connectivity matrix, and the set of constraints $\mathcal{L}_g(\mathbf{A})$ leads to $\mathbf{\Theta}$ being a GGL matrix.

Similarly, the diagonally dominant generalized graph Laplacian (DDGL) estimation problem is formulated as:

$$\min_{\mathbf{\Theta}} \ \operatorname{Tr}(\mathbf{\Theta K}) - \log\det\mathbf{\Theta} \ \text{ subject to } \ \mathbf{\Theta} \in \mathcal{L}_d(\mathbf{A}) \tag{2.34}$$

where there is an additional constraint (*i.e.*, $\mathbf{\Theta}1 \geq 0$) in $\mathcal{L}_d(\mathbf{A})$, which ensures that all node weights are non-negative, and therefore the optimal solution is a diagonally dominant matrix.

Finally, the combinatorial graph Laplacian (CGL) estimation problem is formulated as:

$$\min_{\mathbf{\Theta}} \ \operatorname{Tr}(\mathbf{\Theta K}) - \log|\mathbf{\Theta}| \ \text{ subject to } \ \mathbf{\Theta} \in \mathcal{L}_c(\mathbf{A}) \tag{2.35}$$

where the objective function includes the pseudo-determinant term (*i.e.*, $\log|\mathbf{\Theta}|$), since the solution matrix $\mathbf{\Theta}$ is singular. The pseudo-determinant term makes the optimization problem hard to solve, so (2.35) can be reformulated as follows:

$$\min_{\boldsymbol{\Theta}} \quad \mathrm{Tr}(\boldsymbol{\Theta}(\mathbf{K}+\mathbf{J})) - \log\det(\boldsymbol{\Theta}+\mathbf{J}) \;\; \text{subject to} \;\; \boldsymbol{\Theta} \in \mathcal{L}_c(\mathbf{A}) \tag{2.36}$$

where $\mathbf{J} = \mathbf{u}_1\mathbf{u}_1^\top$ and $\mathbf{u}_1$ is the eigenvector corresponding to the zero eigenvalue of CGL matrix. There is also an additional constraint (*i.e.*, $\boldsymbol{\Theta}1 = 0$) in $\mathcal{L}_c(\mathbf{A})$, which guarantees that the solution is a CGL matrix.

The main idea in their work is to generalize the classical GLASSO formulation to identify GMRF models such that the precision matrix has the form of a graph Laplacian. Estimation of diagonally dominant generalized graph Laplacians (DDGLs) is studied for the first time in [6]. Their method to estimate combinatorial graph Laplacians (CGLs), which is obtained from the novel formulation for the objective function in (2.36), has significant improvement comparing to methods in [9, 58, 84]. They also developed efficient *block-coordinate descent* (BCD) algorithms to solve their optimization problem extended from the algorithm in [85].

In [7], assumptions on spectral properties—*i.e.*, *eigenvalues* of graph Laplacian—is introduced in their graph learning algorithm. Like GLASSO, their problem formulation learns a graph Laplacian matrix, with addition constraints related to eigenvalues of $\mathbf{L}$. Specifically, their optimization framework is as follows:

$$\max_{\boldsymbol{\Theta}} \quad \log \mathrm{gdet}(\boldsymbol{\Theta}) - \mathrm{Tr}(\boldsymbol{\Theta}\mathbf{S}) - \alpha h(\boldsymbol{\Theta}), \qquad \text{s.t.} \;\; \boldsymbol{\Theta} \in \mathcal{S}_{\boldsymbol{\Theta}}, \;\; \lambda(\boldsymbol{\Theta}) \in \mathcal{S}_\lambda \tag{2.37}$$

where $\mathrm{gdet}(\boldsymbol{\Theta})$ is the *generalized determinant* [39], defined as the multiplication of the non-zero eigenvalues of $\boldsymbol{\Theta}$. $\mathbf{S}$ is the sample covariance matrix obtained from data, $\mathcal{S}_{\boldsymbol{\Theta}}$ is the Laplacian matrix structural constraint, $\lambda(\boldsymbol{\Theta})$ is the set of eigenvalues of $\boldsymbol{\Theta}$, and $\mathcal{S}_\lambda$ is the set of spectral constraints on the eigenvalues.

Different choices of $\mathcal{S}_\lambda$ are introduced to enforce learned graphs to have specific structures. For example, *k-component graph* is a graph with a nodes set that can be divided into k separate subsets, such that any two nodes are not connected if they do not belong in same subset. The eigenvalues of these kind of matrices can be represented as follows:

$$\mathcal{S}_\lambda = \{\{\lambda_j = 0\}_{j=1}^{k}, c_1 \leq \lambda_{k+1} \leq ... \leq \lambda_p \leq c_2\} \tag{2.38}$$

where c_1, c_2 are constant parameters depending on the graph edges [86, 87], and k is the number of connected components in the graph.

Further, *k-component d-regular graph* is a k-component graph, where all vertices have degrees equal to d. This also means that the diagonal entries of matrix $\boldsymbol{\Theta}$ equal to d. Thus, to enforce the graph to be k-component d-regular, the following constraint is needed:

$$\mathcal{S}_\lambda = \{\{\lambda_j = 0\}_{j=1}^{k}, c_1 \leq \lambda_{k+1} \leq ... \leq \lambda_p \leq c_2\}, \quad \text{diag}(\Theta) = d\mathbf{1}. \tag{2.39}$$

Finally, learning a *connected sparse graph* is critical for many applications, since promoting sparse connections can lead to a disconnected graph [88, 89]. To learn connected sparse graphs, the following constraints are used:

$$\mathcal{S}_\lambda = \{\lambda_1 = 0, c_1 \leq \lambda_2 \leq ... \leq \lambda_p \leq c_2\} \tag{2.40}$$

where $c_1, c_2 > 0$ are constant parameter, and sparsity is enforced in the nodal domain in addition to constraints on eigenvalues in the spectral domain.

To compare this method with the method in [6] and GLASSO, the perceptual graphs of animal connections are illustrated in Figure 2.5 from [7], where nodes represent animals, and edge weights encode similarity among them.

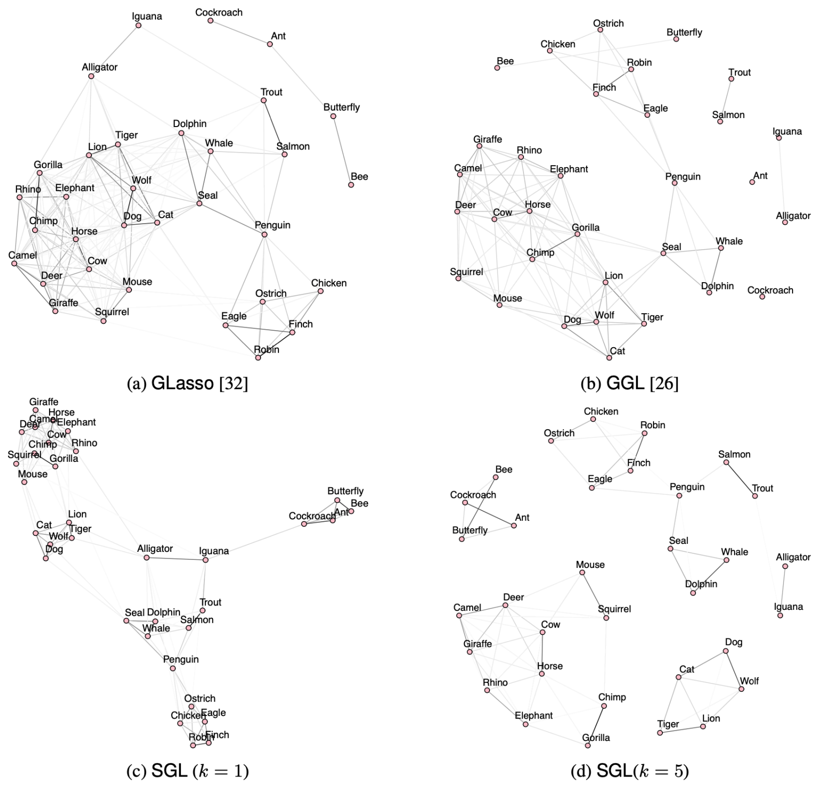


Figure 2.5: (a) GGL [6], **(b)** GLASSO, and **(c)** SGL (proposed method) with components number k = 1, and **(d)** SGL with components number k = 5 [7].

Chapter 3

Convex Cone Projection Operator

In this chapter, a projection operator that projects a *positive definite* (PD), real and symmetric matrix to a convex set of matrices sharing the first K eigenvectors is proposed. In Section 3.1, definition of a Hilbert space for the posed problem is presented. In Section 3.2, the subspace of symmetric *positive semi-definite* (PSD) matrices that share a common *ordered* set of first K eigenvectors is defined, and it is proven that this subspace is a convex cone. In linear algebra, a *convex cone* is a subset of a vector space that is closed under linear combinations with non-negative coefficients [90]. In Section 3.3, an optimization problem to project a PD matrix to the defined convex cone is formulated. Finally, in Section 3.4, the proposed projection operator, inspired by the Gram-Schmidt procedure, [31] is discussed.

3.1 Hilbert Space

Following terminologies in Section 2.2.4, denote by $\mathcal{X}$ a *vector space* of real, symmetric matrices in $\mathbb{R}^{N\times N}$. Note that $\mathcal{X}$ is closed under addition and scalar multiplication. Next, based on the definition in Section 2.2.2, the standard *inner product* [40] $\langle\cdot,\cdot\rangle$

for matrices $\mathbf{P}, \mathbf{Q} \in \mathcal{X}$ is defined as:

$$\langle \mathbf{P}, \mathbf{Q} \rangle = \mathrm{Tr}(\mathbf{Q}^\top \mathbf{P}) = \sum_{i,j} P_{ij} Q_{ij}. \tag{3.1}$$

This definition of inner product induces the Frobenius norm:

$$\langle \mathbf{P}, \mathbf{P} \rangle = \|\mathbf{P}\|_F^2 = \sum_{i,j} P_{ij}^2 \geq 0. \tag{3.2}$$

A *Hilbert space* $\mathcal{H}$ in $\mathcal{X}$ is now defined as a space of real, symmetric matrices endowed with an inner product $\langle \cdot, \cdot \rangle : (\mathcal{H}, \mathcal{H}) \mapsto \mathbb{R}$. Further, the set of *positive semi-definite* (PSD) matrices is defined as

$$\mathcal{H}^+ = \{\mathbf{P} \in \mathcal{H} \,|\, \mathbf{P} \succeq 0\}. \tag{3.3}$$

We state formally and prove $\mathcal{H}^+$ is a convex cone.

Lemma 1. *$\mathcal{H}^+$ is a convex cone.*

Proof. Consider matrices $\mathbf{P}_1, \mathbf{P}_2 \in \mathcal{H}^+$ and non-negative, real scalars $c_1, c_2 \in \mathbb{R}$ where $c_1, c_2 \geq 0$. Consider the linear combination $\mathbf{P} = c_1\mathbf{P}_1 + c_2\mathbf{P}_2$ and its quadratic form:

$$\begin{aligned} \mathbf{x}^\top \mathbf{P} \mathbf{x} &= \mathbf{x}^\top \left(c_1 \mathbf{P}_1 + c_2 \mathbf{P}_2\right) \mathbf{x} && (3.4) \\ &= c_1 \left(\mathbf{x}^\top \mathbf{P}_1 \mathbf{x}\right) + c_2 \left(\mathbf{x}^\top \mathbf{P}_2 \mathbf{x}\right) \overset{(a)}{\geq} 0, \qquad \forall \mathbf{x} \in \mathbb{R}^N && (3.5) \end{aligned}$$

where (a) is true since $\mathbf{P}_1$ and $\mathbf{P}_2$ are PSD, and hence $\mathbf{x}^\top \mathbf{P}_1 \mathbf{x} \geq 0$ and $\mathbf{x}^\top \mathbf{P}_2 \mathbf{x} \geq 0$ for any $\mathbf{x} \in \mathbb{R}^N$. Thus, one can conclude that $\mathbf{P} \in \mathcal{H}^+$, and $\mathcal{H}^+$ is a convex cone [90]. $\square$

3.2 Subspace of Matrices with Common First K Eigenvector

Denote by $\mathcal{H}^+_{\mathbf{u}} \subset \mathcal{H}^+$ the set of PSD matrices that share the first K eigenvectors $\{\mathbf{u}_k\}_{k=1}^K$, assumed to be orthonormal, *i.e.*,

$$\mathbf{u}_j^\top \mathbf{u}_k = \delta_{j-k}, \quad \forall j, k \in \mathcal{I}_K \tag{3.6}$$

where $\mathcal{I}_K = \{1, \ldots, K\}$, and δ_i is the discrete impulse function that evaluates to 1 if $i = 0$, and 0 otherwise. $\mathcal{H}^+_{\mathbf{u}}$ can be defined using the *Rayleigh quotient* and the min-max theorem [91] as

$$\mathcal{H}^+_{\mathbf{u}} = \left\{ \mathbf{L} \in \mathcal{H}^+ \mid \mathbf{u}_k = \arg \min_{\mathbf{x} \,|\, \mathbf{x} \perp \mathbf{u}_j, \forall j < k} \frac{\mathbf{x}^\top \mathbf{L} \mathbf{x}}{\mathbf{x}^\top \mathbf{x}}, \quad k \in \mathcal{I}_K \right\}. \tag{3.7}$$

where $\frac{\mathbf{x}^\top \mathbf{L} \mathbf{x}}{\mathbf{x}^\top \mathbf{x}}$ is the Rayleigh quotient for vector $\mathbf{x}$.

It can be shown that, for a given real symmetric matrix, the Rayleigh quotient reaches its minimum value $\lambda_{\min}$ (the smallest eigenvalue of $\mathbf{L}$) when $\mathbf{x}$ is $\mathbf{v}_{\min}$ (the corresponding eigenvector) [92].

We show next that $\mathcal{H}^+_{\mathbf{u}}$ is also a convex cone.

Lemma 2. *$\mathcal{H}^+_{\mathbf{u}}$ is a convex cone.*

Proof. This lemma can be proven by induction. For the base case when $K = 1$, let $\mathbf{L}_1, \mathbf{L}_2 \in \mathcal{H}^+$ be two matrices that share the first eigenvector $\mathbf{u}_1$. Consider a linear combination $\mathcal{L} = c_1 \mathbf{L}_1 + c_2 \mathbf{L}_2$, where $c_1, c_2 \geq 0$. The smallest eigenvalue $\lambda_{\min}(\mathcal{L})$ of $\mathcal{L}$

can be expressed using the Rayleigh quotient again, *i.e.*,

$$\lambda_{\min}(\mathcal{L}) = \min_{\mathbf{x}} \frac{\mathbf{x}^\top \mathcal{L} \mathbf{x}}{\mathbf{x}^\top \mathbf{x}} = \min_{\mathbf{x}} \frac{\mathbf{x}^\top \left(c_1 \mathbf{L}_1 + c_2 \mathbf{L}_2\right) \mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \tag{3.8}$$

$$\overset{(a)}{\geq} c_1 \left(\min_{\mathbf{x}} \frac{\mathbf{x}^\top \mathbf{L}_1 \mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \right) + c_2 \left(\min_{\mathbf{y}} \frac{\mathbf{y}^\top \mathbf{L}_2 \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \right) \tag{3.9}$$

$$\overset{(b)}{=} c_1 \, \lambda_{\min}(\mathbf{L}_1) + c_2 \, \lambda_{\min}(\mathbf{L}_2). \tag{3.10}$$

The inequality at (a) is true since the right-hand side (RHS) has more degrees of freedom than the left-hand side (LHS) to minimize the two non-negative terms ($\mathbf{L}_1$ and $\mathbf{L}_2$ are PSD matrices in $\mathcal{H}^+$). (b) is true by definition of Rayleigh quotient for $\mathbf{L}_1$ and $\mathbf{L}_2$, with $\mathbf{u}_1$ being the minimizing argument for both terms by assumption. Thus, the argument that minimizes the Rayleigh quotient for $\mathcal{L}$ is also $\mathbf{u}_1$, and therefore $\mathbf{u}_1$ is the first eigenvector of $\mathcal{L}$. Since this analysis is true for all $c_1, c_2 \geq 0$, $\mathcal{H}^+_{\mathbf{u}}$ is a convex cone for $K = 1$.

Consider next the inductive step, where it is assumed that the $K = K_o$ case is true, and the $K_o{+}1$ case is examined. Let $\mathbf{L}_1, \mathbf{L}_2 \in \mathcal{H}^+$ be two matrices that share the first $K_o + 1$ eigenvectors $\{\mathbf{u}_k\}_{k=1}^{K_o+1}$. Consider a linear combination $\mathcal{L} = c_1 \mathbf{L}_1 + c_2 \mathbf{L}_2$, where $c_1, c_2 \geq 0$. By the inductive assumption, $\mathcal{L}$ also shares the first K_o eigenvectors $\{\mathbf{u}_k\}_{k=1}^{K_o}$. For the $(K_o{+}1)$-th eigenvector, consider the $(K_o{+}1)$-th eigenvalue $\lambda_{K_o+1}(\mathcal{L})$ expressed again using the Rayleigh quotient as:

$$\lambda_{K_o+1}(\mathcal{L}) = \min_{\mathbf{x} | \mathbf{x} \perp \{\mathbf{u}_k\}_{k=1}^{K}} \frac{\mathbf{x}^\top \mathcal{L} \mathbf{x}}{\mathbf{x}^\top \mathbf{x}} = \min_{\mathbf{x} | \mathbf{x} \perp \{\mathbf{u}_k\}_{k=1}^{K}} \frac{\mathbf{x}^\top \left(c_1 \mathbf{L}_1 + c_2 \mathbf{L}_2\right) \mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \tag{3.11}$$

$$\geq c_1 \left(\min_{\mathbf{x} | \mathbf{x} \perp \{\mathbf{u}_k\}_{k=1}^{K}} \frac{\mathbf{x}^\top \mathbf{L}_1 \mathbf{x}}{\mathbf{x}^\top \mathbf{x}} \right) + c_2 \left(\min_{\mathbf{y} | \mathbf{y} \perp \{\mathbf{u}_k\}_{k=1}^{K}} \frac{\mathbf{y}^\top \mathbf{L}_2 \mathbf{y}}{\mathbf{y}^\top \mathbf{y}} \right) \tag{3.12}$$

$$= c_1 \, \lambda_{K_o+1}(\mathbf{L}_1) + c_2 \lambda_{K_o+1}(\mathbf{L}_2). \tag{3.13}$$

Since the argument that minimizes the Rayleigh quotient for both $\mathbf{L}_1$ and $\mathbf{L}_2$ is $\mathbf{u}_{K_o+1}$, this is also the argument that minimizes the Rayleight quotient for $\mathcal{L}$. Thus, the $(K_o + 1)$-th eigenvector for $\mathcal{L}$ is also $\mathbf{u}_{K_o+1}$. Therefore, it can be concluded that $\mathcal{L}$ has $\{\mathbf{u}_k\}_{k=1}^{K_o+1}$ as its first $K_o + 1$ eigenvectors. Since both the base case and the inductive step are true, $\mathcal{H}_{\mathbf{u}}^+$ is a convex cone. $\square$

3.3 Projection to Convex Cone $\mathcal{H}_{\mathbf{u}}^+$

To project a PD matrix $\mathbf{P} \in \mathcal{H}^+$ to convex cone $\mathcal{H}_{\mathbf{u}}^+$, the focus is on the inverse $\mathbf{C} = \mathbf{P}^{-1}$ instead. The reason is that it is easy to first ensure that $\mathbf{u}_1$ of the known eigenvector set $\{\mathbf{u}_k\}_{k=1}^{K}$ is the argument that *maximizes* the Rayleigh quotient of $\mathbf{C}$, then $\mathbf{u}_2$ is the argument that maximizes the Rayleigh quotient while being orthogonal to $\mathbf{u}_1$, and so on until $\mathbf{u}_K$.

Consider first the maximization of the Rayleigh quotient of $\mathbf{C}$ with argument $\mathbf{u}_1$. First, define the subspace spanned by rank-1 matrix $\mathbf{U}_1 = \mathbf{u}_1\mathbf{u}_1^\top$ with scalar α as follows:

$$\boldsymbol{\Omega} = \{\alpha \mathbf{U}_1, \alpha \in \mathbb{R}\}. \tag{3.14}$$

Projection of any real symmetric matrix $\mathbf{M} \in \mathcal{H}$ onto 1-dimension subspace $\boldsymbol{\Omega}$ is $\text{Proj}_{\boldsymbol{\Omega}}(\mathbf{M}) = \langle \mathbf{M}, \mathbf{U}_1 \rangle \mathbf{U}_1$ (Example 2.22 in [1]). It is easy to verify that $\langle \mathbf{M}, \mathbf{U}_1 \rangle \mathbf{U}_1$ is a linear operator (satisfying distributivity and linearity in the first argument of the inner product) that is idempotent, and thus $\langle \mathbf{M}, \mathbf{U}_1 \rangle \mathbf{U}_1$ is a projection. One can also prove that $\langle \mathbf{M}, \mathbf{U}_1 \rangle \mathbf{U}_1$ is self-adjoint, and thus $\langle \mathbf{M}, \mathbf{U}_1 \rangle \mathbf{U}_1$ is an orthogonal projection.

Denote by $\boldsymbol{\Omega}^\perp$ the orthogonal subspace in $\mathcal{H}$ so that $\mathcal{H} = \boldsymbol{\Omega} \oplus \boldsymbol{\Omega}^\perp$; $\boldsymbol{\Omega}$ is the *range*

$\mathcal{R}(\text{Proj}_{\mathbf{\Omega}}(\cdot))$ of projection operator $\text{Proj}_{\mathbf{\Omega}}(\cdot)$, and $\mathbf{\Omega}^{\perp}$ is the *null space* $\mathcal{N}(\text{Proj}_{\mathbf{\Omega}}(\cdot))$ of $\text{Proj}_{\mathbf{\Omega}}(\cdot)$. This is called *subspace decomposition*, which is always possible using a projection operator (Theorem 2.32 in [1]).

First, $\mathbf{C}$ is written as its projection onto $\mathbf{\Omega}$ plus its orthogonal component, *i.e.*,

$$\mathbf{C} = \langle \mathbf{C}, \mathbf{U}_1 \rangle \mathbf{U}_1 \; + \; \mathbf{C}_{\mathbf{\Omega}^{\perp}}, \tag{3.15}$$

where $\mathbf{C}_{\mathbf{\Omega}^{\perp}}$ is the component of $\mathbf{C}$ in subspace $\mathbf{\Omega}^{\perp}$, orthogonal to component $\langle \mathbf{C}, \mathbf{U}_1 \rangle \mathbf{U}_1$ in $\mathbf{\Omega}$. (3.15) is called an *orthogonal decomposition* of $\mathbf{C} \in \mathcal{H}$ (Definition 2.31 in [1]).

It can be shown that given $\mathbf{C}$ is PD, inner product $\langle \mathbf{C}, \mathbf{U} \rangle$ for any outer product $\mathbf{U} = \mathbf{u}\mathbf{u}^{\top}$ is non-negative. This is formalized in the following lemma.

Lemma 3. *If* $\mathbf{C} \in \mathcal{H}$ *is PD and* $\mathbf{U} = \mathbf{u}\mathbf{u}^{\top}$, *then* $\langle \mathbf{C}, \mathbf{U} \rangle \geq 0$.

Proof. Since $\mathbf{C} \in \mathcal{H}$ is real and symmetric and is PD by assumption, without loss of generality, $\mathbf{C} = \mathbf{B}\mathbf{B}^{\top}$ can be written via Cholesky factorization [93]. Then, based on the definition of inner product in Section 2.2.2, $\langle \mathbf{C}, \mathbf{U} \rangle$ can be rewritten as:

$$\begin{aligned} \langle \mathbf{C}, \mathbf{U} \rangle &= \text{Tr}(\mathbf{U}^{\top}\mathbf{C}) = \text{Tr}\left((\mathbf{u}\mathbf{u}^{\top})^{\top}\mathbf{B}\mathbf{B}^{\top}\right) \\ &= \text{Tr}\left(\mathbf{u}\mathbf{u}^{\top}\mathbf{B}\mathbf{B}^{\top}\right) \overset{(a)}{=} \text{Tr}\left(\mathbf{u}^{\top}\mathbf{B}\mathbf{B}^{\top}\mathbf{u}\right) \\ &= \text{Tr}\left((\mathbf{B}^{\top}\mathbf{u})^{\top}\mathbf{B}^{\top}\mathbf{u}\right) = \langle \mathbf{B}^{\top}\mathbf{u}, \mathbf{B}^{\top}\mathbf{u} \rangle = \|\mathbf{B}^{\top}\mathbf{u}\|_F^2 \geq 0 \end{aligned} \tag{3.16}$$

where (a) is true since trace of a sequence of product terms is invariant to cyclic permutation of the terms. The last term is a Frobenius norm $\|\mathbf{B}^{\top}\mathbf{u}\|_F^2 \geq 0$. □

Define $\mu_1 = \langle \mathbf{C}, \mathbf{U}_1 \rangle \geq 0$. $\mathbf{\Omega}^{\perp}$ can be expressed as the span of mutually orthogonal rank-1 matrices $\mathbf{U}_2, \ldots, \mathbf{U}_K$ and $\mathbf{V}_{K+1}, \ldots, \mathbf{V}_N$, each of which is orthogonal to $\mathbf{U}_1$, where $\mathbf{U}_k = \mathbf{u}_k\mathbf{u}_k^\top$, $\mathbf{V}_i = \mathbf{v}_i\mathbf{v}_i^\top$ and $\|\mathbf{v}_i\|_2 = 1$. Thus, rank-1 matrices $\{\mathbf{U}_k\}_{k=1}^K$ and $\{\mathbf{V}_i\}_{i=K+1}^N$ collectively form an *orthonormal basis* for Hilbert space $\mathcal{H}$ (Definition 2.38 in [1]). Hence, by Theorem 2.39 in [1], we can write $\mathbf{C}$ as an *orthonormal basis expansion* of these bases, *i.e.*,

$$\mathbf{C} = \sum_{k=1}^{K} \langle \mathbf{C}, \mathbf{U}_k \rangle \mathbf{U}_k + \sum_{i=K+1}^{N} \langle \mathbf{C}, \mathbf{V}_i \rangle \mathbf{V}_i. \tag{3.17}$$

Suppose that $\mathbf{u}_k$'s and $\mathbf{v}_i$'s are successive arguments that maximize the Rayleigh quotient, *i.e.*,

$$\begin{aligned} \mathbf{u}_k &= \arg\max_{\mathbf{u}} \frac{\mathbf{u}^\top \mathbf{C}\mathbf{u}}{\mathbf{u}^\top \mathbf{u}}, \qquad \text{s.t. } \mathbf{u}_j^\top \mathbf{u} = 0, \quad \forall j \in \{1, \ldots, k-1\} \\ \mathbf{v}_i &= \arg\max_{\mathbf{v}} \frac{\mathbf{v}^\top \mathbf{C}\mathbf{v}}{\mathbf{v}^\top \mathbf{v}}, \qquad \text{s.t. } \begin{cases} \mathbf{u}_j^\top \mathbf{v} = 0, & \forall j \in \{1, \ldots, K\} \\ \mathbf{v}_j^\top \mathbf{v} = 0, & \forall j \in \{K+1, \ldots, i-1\} \end{cases}. \end{aligned} \tag{3.18}$$

(3.18) implies that $\{\mathbf{u}_k\}_{k=1}^K$ and $\{\mathbf{v}_i\}_{i=K+1}^N$ are the eigenvectors of $\mathbf{C}$.

Suppose also that the inner products of $\mathbf{C}$ with the orthogonal components, $\langle \mathbf{C}, \mathbf{U}_k \rangle$, $\langle \mathbf{C}, \mathbf{V}_i \rangle$, are within range $[0, \mu_1]$, $\forall k, i$. This means that the respective Rayleigh quotients are no larger than μ_1:

$$\begin{aligned} \mathrm{Tr}(\mathbf{u}_k^\top \mathbf{C}\mathbf{u}_k) &= \mathrm{Tr}(\mathbf{C}\mathbf{u}_k\mathbf{u}_k^\top) = \langle \mathbf{C}, \mathbf{U}_k \rangle \leq \mu_1, \qquad \forall k \in \{2, \ldots, K\} \\ \mathrm{Tr}(\mathbf{v}_i^\top \mathbf{C}\mathbf{v}_i) &= \mathrm{Tr}(\mathbf{C}\mathbf{v}_i\mathbf{v}_i^\top) = \langle \mathbf{C}, \mathbf{V}_i \rangle \leq \mu_1, \qquad \forall i \in \{K+1, \ldots, N\}. \end{aligned} \tag{3.19}$$

Given these conditions, then $\mathbf{u}_1$ is surely the *last* eigenvector of $\mathbf{C}$ corresponding to *largest* eigenvalue μ_1. More generally, one needs to approximate $\mathbf{C}$'s projection $\mathbf{C}_{\mathbf{\Omega}^\perp}$ to $\mathbf{\Omega}^\perp$ as $\hat{\mathbf{C}}_{\mathbf{\Omega}^\perp}$ to satisfy these conditions, in order to ensure $\mathbf{u}_1$ is indeed the last eigenvector of an approximation $\hat{\mathbf{C}} = \langle \mathbf{C}, \mathbf{U}_1 \rangle \mathbf{U}_1 + \hat{\mathbf{C}}_{\mathbf{\Omega}^\perp}$, while minimizing approximation error $\|\mathbf{C} - \hat{\mathbf{C}}\|_F^2$. A greedy algorithm is proposed for this approximation next.

3.4 Gram-Schmidt-inspired Algorithm

The greedy algorithm is inspired by the *Gram-Schmidt* procedure [31] that iteratively computes a set of orthonormal vectors given a set of linearly independent vectors. Similarly, this algorithm iteratively computes one orthogonal eigen-component $\mu_k \mathbf{U}_k$ or $\mu_i \mathbf{V}_i$ at a time. Note again that $\mathbf{V}_i = \mathbf{v}_i \mathbf{v}_i^\top$ must be computed, while $\mathbf{U}_k = \mathbf{u}_k \mathbf{u}_k^\top$ is known *a priori*.

Consider iteration $t = 2$. From (3.15), the *residual signal* is first defined as:

$$\mathbf{E}_1 = \mathbf{C} - \langle \mathbf{C}, \mathbf{U}_1 \rangle \mathbf{U}_1. \tag{3.20}$$

Consider first the case where $K \geq 2$, and thus rank-1 matrix $\mathbf{U}_2 = \mathbf{u}_2 \mathbf{u}_2^\top$ orthogonal to $\mathbf{U}_1$ is known. The only requirement for eigen-component $\mu_2 \mathbf{U}_2$ is that the inner product $\mu_2 = \langle \hat{\mathbf{C}}, \mathbf{U}_2 \rangle$ of approximation $\hat{\mathbf{C}}$ and $\mathbf{U}_2$ is no larger than μ_1. Thus, μ_2 can be set as

$$\mu_2 = \begin{cases} \langle \mathbf{E}_1, \mathbf{U}_2 \rangle, & \text{if } \langle \mathbf{E}_1, \mathbf{U}_2 \rangle \leq \mu_1 \\ \mu_1, & \text{o.w.} \end{cases}. \tag{3.21}$$

Note that $\langle \mathbf{E}_1, \mathbf{U}_2 \rangle \geq 0$. Lemma 5, to be formalized later, describes the more general case $\langle \mathbf{E}_i, \mathbf{U}_{i+1} \rangle \geq 0$, for $1 \leq i \leq K - 1$.

Consider next the case where $K = 1$, and a new rank-1 matrix $\mathbf{V}_2 = \mathbf{v}_2 \mathbf{v}_2^\top$ must be computed—one that is orthogonal to $\mathbf{U}_1$—to reconstruct $\hat{\mathbf{C}}$. In this case, to minimize approximation error $\|\mathbf{C}-\hat{\mathbf{C}}\|_F^2$, a $\mathbf{V}_2$ "most aligned" with residual signal $\mathbf{E}_1$ is sought, *i.e.*,

$$\max_{\mathbf{v}_2} \; \langle \mathbf{E}_1, \mathbf{v}_2\mathbf{v}_2^\top \rangle, \quad \text{s.t.} \quad \begin{cases} \langle \mathbf{v}_2\mathbf{v}_2^\top, \mathbf{U}_1 \rangle = 0 \\ \|\mathbf{v}_2\|_2 = 1 \end{cases} . \tag{3.22}$$

Essentially, (3.22) seeks a *rank-1 approximation* $\langle \mathbf{E}_1, \mathbf{V}_2 \rangle \mathbf{V}_2$ of matrix $\mathbf{E}_1$, while constraining $\mathbf{V}_2$ to be orthogonal to $\mathbf{U}_1$. The objective is equivalent to $\text{Tr}((\mathbf{v}_2\mathbf{v}_2^\top)^\top \mathbf{E}_1) = \text{Tr}(\mathbf{v}_2\mathbf{v}_2^\top \mathbf{E}_1) = \text{Tr}(\mathbf{v}_2^\top \mathbf{E}_1 \mathbf{v}_2) = \mathbf{v}_2^\top \mathbf{E}_1 \mathbf{v}_2$, which is quadratic in $\mathbf{v}_2$ for a maximization problem. It can be shown that $\mathbf{v}_2^\top \mathbf{E}_1 \mathbf{v}_2$ is always convex, *i.e.*, $\mathbf{E}_1$ is PSD. This is formalized as Lemma 4 later in the section for the general case where $\mathbf{E}_t$ is shown to be PSD. This means (3.22) is a non-convex optimization problem.

To convexify the problem, one can first perform approximation $\mathbf{E}_1 \approx \mathbf{e}\mathbf{e}^\top$, where $\mathbf{e}$ is the last eigenvector[1] of $\mathbf{E}_1$, and $\mathbf{e}\mathbf{e}^\top$ is the best rank-1 approximation of $\mathbf{E}_1$. Then, the norm equality constraint can be relaxed, and optimization (3.22) can be reformulated as

$$\max_{\mathbf{v}_2} \mathbf{e}^\top \mathbf{v}_2, \quad \text{s.t.} \quad \begin{cases} \mathbf{v}_2^\top \mathbf{u}_1 = 0 \\ \|\mathbf{v}_2\|_2^2 \leq 1 \end{cases} . \tag{3.23}$$

Optimization (3.23) is now convex and solvable in polynomial time, using algorithms such as *proximal gradient* (PG) [95].

Having computed $\mathbf{V}_2 = \mathbf{v}_2\mathbf{v}_2^\top$ in (3.23), one can project $\mathbf{E}_1$ onto $\mathbf{V}_2$ and threshold

[1]Extreme eigenvectors of sparse symmetric matrices can be computed efficiently using *Locally Optimal Block Preconditioned Conjugate Gradient* (LOBPCG) [94].

its inner product to within $[0, \mu_1]$ as done in the $K \geq 2$ case, *i.e.*,

$$\mu_2 = \begin{cases} \langle \mathbf{E}_1, \mathbf{V}_2 \rangle, & \text{if } \langle \mathbf{E}_1, \mathbf{V}_2 \rangle \leq \mu_1 \\ \mu_1, & \text{o.w.} \end{cases} . \tag{3.24}$$

Note that $\langle \mathbf{E}_1, \mathbf{V}_2 \rangle \geq 0$. Lemma 5, to be formalized later, describes the more general case $\langle \mathbf{E}_i, \mathbf{V}_{i+1} \rangle \geq 0$, for $K \leq i \leq N-1$.

The projection of $\mathbf{E}_1$ on $\mathbf{V}_2$ is thus $\mu_2 \mathbf{V}_2$. New residual signal $\mathbf{E}_2$ can then be computed as:

$$\mathbf{E}_2 = \mathbf{E}_1 - \mu_2 \mathbf{V}_2. \tag{3.25}$$

Given residual $\mathbf{E}_2$, one can compute a rank-1 matrix $\mathbf{V}_3 = \mathbf{v}_3 \mathbf{v}_3^\top$—orthogonal to $\mathbf{V}_2$ and $\mathbf{U}_1$—that is most aligned with $\mathbf{E}_2$, and compute projection of $\mathbf{E}_2$ to $\mathbf{V}_3$, and so on.

More generally, at each iteration $t \leq K$, the inner product is thresholded as

$$\mu_{t+1} = \min(\langle \mathbf{E}_t, \mathbf{U}_{t+1} \rangle, \mu_t). \tag{3.26}$$

Then, the next residual signal is computed as

$$\mathbf{E}_{t+1} = \mathbf{E}_t - \mu_{t+1} \mathbf{U}_{t+1}. \tag{3.27}$$

On the other hand, at each iteration $t \geq K+1$, rank-1 approximation $\mathbf{E}_t \approx \mathbf{e}\mathbf{e}^\top$ is first computed, where $\mathbf{e}$ is the last eigenvector of $\mathbf{E}_t$. Then an optimal $\mathbf{v}_{t+1}$ is computed via convex optimization

$$\max_{\mathbf{v}_{t+1}} \ \mathbf{e}^\top \mathbf{v}_{t+1}, \quad \text{s.t.} \ \begin{cases} \mathbf{v}_{t+1}^\top \mathbf{u}_k = 0 \ \ , i \in \mathcal{I}_K \\ \mathbf{v}_{t+1}^\top \mathbf{v}_\tau = 0 \ \ , \ \ \tau \in \{K+1, \ldots, t\} \\ \|\mathbf{v}_{t+1}\|_2^2 \leq 1 \end{cases} . \tag{3.28}$$

The inner product is then thresholded as

$$\mu_{t+1} = \min(\langle \mathbf{E}_t, \mathbf{V}_{t+1} \rangle, \mu_t) \tag{3.29}$$

where $\mathbf{V}_{t+1} = \mathbf{v}_{t+1}\mathbf{v}_{t+1}^\top$. The next residual signal is computed as

$$\mathbf{E}_{t+1} = \mathbf{E}_t - \mu_{t+1}\mathbf{V}_{t+1}. \tag{3.30}$$

This procedure is repeated until the last eigen-component $\mu_N \mathbf{V}_N$ is computed, and the approximation of $\mathbf{C}$ is $\hat{\mathbf{C}} = \sum_{k=1}^K \mu_k \mathbf{U}_k + \sum_{i=K+1}^N \mu_i \mathbf{V}_i$. Collectively, the iterative procedure forms the projection operator $\text{Proj}_{\mathcal{H}_\mathbf{u}^+}(\mathbf{P})$ for PD matrix $\mathbf{P}$ into cone $\mathcal{H}_\mathbf{u}^+$, albeit it operates entirely on inverse matrix $\mathbf{C} = \mathbf{P}^{-1}$.

Having described the iterative procedure, lemma stating residual energy $\mathbf{E}_t$ is PSD is now formalized.

Lemma 4. *Residual energy $\mathbf{E}_t$ is PSD, for $t \in \{1, \ldots, N\}$.*

Proof. First, one can write

$$\mathbf{E}_t = \mathbf{E}_{t-1} - \mu_t \mathbf{U}_t \tag{3.31}$$

$$= \mathbf{E}_{t-2} - \mu_{t-1}\mathbf{U}_{t-1} - \mu_t \mathbf{U}_t \tag{3.32}$$

$$= \mathbf{C} - \sum_{i=1}^{t} \mu_i \mathbf{U}_i. \tag{3.33}$$

Recall that $\mu_1 = \langle \mathbf{C}, \mathbf{U}_1 \rangle$ and $\mu_{t+1} = \min(\langle \mathbf{E}_t, \mathbf{U}_{t+1} \rangle, \mu_t)$ for $t \geq 1$. Denote by $\boldsymbol{\Omega}$ the subspace in $\mathcal{H}$ spans by t orthogonal rank-1 matrices $\{\mathbf{U}_i\}_{i=1}^t$, *i.e.*, $\boldsymbol{\Omega} = \{\mathbf{M} \mid \mathbf{M} = \sum_{i=1}^t \alpha_i \mathbf{U}_i,\ \forall \alpha_i \in \mathbb{R}\}$. Denote by $\boldsymbol{\Omega}^\perp$ the subspace in $\mathcal{H}$ orthogonal to $\boldsymbol{\Omega}$, where $\mathcal{H} = \boldsymbol{\Omega} \oplus \boldsymbol{\Omega}^\perp$. One can now write $\mathbf{C} \in \mathcal{H}$ as

$$\mathbf{C} = \sum_{i=1}^{t} \gamma_i \mathbf{U}_i + \mathbf{C}_{\Omega^\perp} \tag{3.34}$$

where $\gamma_i = \langle \mathbf{C}, \mathbf{U}_i \rangle$, and $\mathbf{C}_{\boldsymbol{\Omega}^\perp}$ is the projection of $\mathbf{C}$ to orthogonal subspace $\boldsymbol{\Omega}^\perp$. Thus, one can now write

$$\mathbf{x}^\top \mathbf{E}_t \mathbf{x} = \mathbf{x}^\top \left(\mathbf{C} - \sum_{i=1}^{t} \mu_i \mathbf{U}_i \right) \mathbf{x} \tag{3.35}$$

$$= \mathbf{x}^\top \left(\sum_{i=1}^{t} (\gamma_i - \mu_i) \mathbf{U}_i \right) \mathbf{x} + \mathbf{x}^\top \mathbf{C}_{\boldsymbol{\Omega}^\perp} \mathbf{x} \tag{3.36}$$

$$\overset{(a)}{=} \sum_{i=1}^{t} (\gamma_i - \mu_i) |\mathbf{u}_i^\top \mathbf{x}|^2 + \mathbf{x}^\top \mathbf{C}_{\boldsymbol{\Omega}^\perp} \mathbf{x} \tag{3.37}$$

where (a) is true since $\mathbf{x}^\top \mathbf{U}_i \mathbf{x} = \mathbf{x}^\top \mathbf{u}_i \mathbf{u}_i^\top \mathbf{x} = |\mathbf{u}_i^\top \mathbf{x}|^2$. Thus, one can show $\mathbf{E}_t$ is PSD by showing i) $\gamma_i \geq \mu_i, \forall i \in \{1, \ldots, t\}$, and ii) $\mathbf{C}_{\Omega^\perp}$ is PSD.

Because $\mu_i = \min(\langle \mathbf{E}_{i-1}, \mathbf{U}_i \rangle, \mu_{i-1})$ and $\gamma_i = \langle \mathbf{C}, \mathbf{U}_i \rangle$, we can write

$$\mu_i \overset{(a)}{=} \min(\langle \mathbf{C} - \sum_{j=1}^{i-1} \mu_j \mathbf{U}_j, \mathbf{U}_i \rangle, \mu_{i-1}) \tag{3.38}$$

$$\overset{(b)}{=} \min(\langle \mathbf{C}, \mathbf{U}_i \rangle, \mu_{i-1}) \tag{3.39}$$

$$= \min(\gamma_i, \mu_{i-1}) \leq \gamma_i \tag{3.40}$$

where (a) follows from (3.33), and (b) is due to the orthogonality of $\{\mathbf{U}_i\}_{i=1}^K$.

To show $\mathbf{C}_{\mathbf{\Omega}^\perp}$ is PSD, note first that $\mathbf{\Omega}^\perp$ can be expressed as the span of orthogonal $\mathbf{U}_{t+1}, \ldots, \mathbf{U}_K$ and $\mathbf{V}_{K+1}, \ldots, \mathbf{V}_N$. Thus, $\mathbf{C}_{\mathbf{\Omega}^\perp} \in \Omega^\perp$ can be written as:

$$\mathbf{C}_{\mathbf{\Omega}^\perp} = \sum_{k=t+1}^{K} \langle \mathbf{C}, \mathbf{U}_k \rangle \mathbf{U}_k + \sum_{i=K+1}^{N} \langle \mathbf{C}, \mathbf{V}_i \rangle \mathbf{V}_i \tag{3.41}$$

From Lemma 3, given $\mathbf{C}$ is PD, $\langle \mathbf{C}, \mathbf{U}_k \rangle \geq 0, \forall k$ and $\langle \mathbf{C}, \mathbf{V}_i \rangle \geq 0, \forall i$. Thus, for any $\mathbf{x} \in \mathbb{R}^N$,

$$\mathbf{x}^\top \mathbf{C}_{\Omega^\perp} \mathbf{x} = \sum_{k=t+1}^{K} \langle \mathbf{C}, \mathbf{U}_k \rangle \mathbf{x}^\top \mathbf{U}_k \mathbf{x} + \sum_{i=K+1}^{N} \langle \mathbf{C}, \mathbf{V}_i \rangle \mathbf{x}^\top \mathbf{V}_i \mathbf{x} \tag{3.42}$$

$$\overset{(a)}{=} \sum_{k=t+1}^{K} \langle \mathbf{C}, \mathbf{U}_k \rangle |\mathbf{u}_k^\top \mathbf{x}|^2 + \sum_{i=K+1}^{N} \langle \mathbf{C}, \mathbf{V}_i \rangle |\mathbf{v}_i^\top \mathbf{x}|^2 \overset{(b)}{\geq} 0 \tag{3.43}$$

where (a) is true since $\mathbf{x}^\top \mathbf{U}_k \mathbf{x} = \mathbf{x}^\top \mathbf{u}_k \mathbf{u}_k^\top \mathbf{x} = |\mathbf{u}_k^\top \mathbf{x}|^2$, and (b) is true since each term in each of the two summations is non-negative. $\square$

Lemma showing the general case of $\langle \mathbf{E}_t, \mathbf{U}_{t+1} \rangle \geq 0$ and $\langle \mathbf{E}_t, \mathbf{V}_{t+1} \rangle \geq 0$ is now formalized.

Lemma 5. *The inner product of signal residual $\mathbf{E}_t$ with rank-1 matrix $\mathbf{U}_{t+1}$ or $\mathbf{V}_{t+1}$*

is non-negative, i.e.,

$$\langle \mathbf{E}_t, \mathbf{U}_{t+1} \rangle \geq 0, \quad \forall t \in \{1, \ldots, K-1\} \tag{3.44}$$

$$\langle \mathbf{E}_t, \mathbf{V}_{t+1} \rangle \geq 0, \quad \forall t \in \{K, \ldots, N-1\}. \tag{3.45}$$

Proof. Consider first the case $t < K$ and (3.44). By definition, $\mathbf{E}_t$'s for $t < K$ are

$$\begin{array}{ll} \mathbf{E}_1 = \mathbf{C} - \mu_1 \mathbf{U}_1, & \mu_1 = \langle \mathbf{C}, \mathbf{U}_1 \rangle \\ \mathbf{E}_2 = \mathbf{E}_1 - \mu_2 \mathbf{U}_2, & \mu_2 = \min(\langle \mathbf{E}_1, \mathbf{U}_2 \rangle, \mu_1) \\ \vdots & \\ \mathbf{E}_{t+1} = \mathbf{E}_t - \mu_{t+1} \mathbf{U}_{t+1}, & \mu_{t+1} = \min(\langle \mathbf{E}_t, \mathbf{U}_{t+1} \rangle, \mu_t) \end{array} . \tag{3.46}$$

$\langle \mathbf{E}_t, \mathbf{U}_{t+1} \rangle \geq 0$ in (3.44) is first proven.

Examining $\langle \mathbf{E}_t, \mathbf{U}_{t+1} \rangle$,

$$\langle \mathbf{E}_t, \mathbf{U}_{t+1} \rangle \overset{(a)}{=} \langle \mathbf{C} - \sum_{i=1}^{t} \mu_i \mathbf{U}_i, \mathbf{U}_{t+1} \rangle \tag{3.47}$$

$$= \langle \mathbf{C}, \mathbf{U}_{t+1} \rangle - \sum_{i=1}^{t} \mu_i \langle \mathbf{U}_i, \mathbf{U}_{t+1} \rangle \tag{3.48}$$

$$\overset{(b)}{=} \langle \mathbf{C}, \mathbf{U}_{t+1} \rangle \overset{(c)}{\geq} 0 \tag{3.49}$$

where (a) is due to (3.33), (b) is true since $\mathbf{U}_{t+1}$ is orthogonal to $\{\mathbf{U}_i\}_{i=1}^{t}$ by definition, and (c) is true due to Lemma 3.

Next $\langle \mathbf{E}_t, \mathbf{V}_{t+1} \rangle \geq 0$ in (3.45) is proven. Similarly, one can write

$$\mathbf{E}_t = \mathbf{C} - \sum_{i=1}^{K} \mu_i \mathbf{U}_i - \sum_{i=K+1}^{t} \mu_i \mathbf{V}_i. \tag{3.50}$$

Examining $\langle \mathbf{E}_t, \mathbf{V}_{t+1} \rangle$,

$$\begin{aligned}
\langle \mathbf{E}_t, \mathbf{V}_{t+1} \rangle &= \langle \mathbf{C} - \sum_{i=1}^{K} \mu_i \mathbf{U}_i - \sum_{i=K+1}^{t} \mu_i \mathbf{V}_i, \mathbf{V}_{t+1} \rangle & (3.51) \\
&= \langle \mathbf{C}, \mathbf{V}_{t+1} \rangle - \sum_{i=1}^{K} \mu_i \langle \mathbf{U}_i, \mathbf{V}_{t+1} \rangle - \sum_{i=K+1}^{t} \mu_i \langle \mathbf{V}_i, \mathbf{V}_{t+1} \rangle & (3.52) \\
&\overset{(a)}{=} \langle \mathbf{C}, \mathbf{V}_{t+1} \rangle \overset{(b)}{\geq} 0 & (3.53)
\end{aligned}$$

where (a) is true since $\mathbf{V}_{t+1}$ is orthogonal to $\{\mathbf{U}_i\}_{i=1}^K$ and $\{\mathbf{V}_i\}_{i=K+1}^t$ by construction, and (b) is true by Lemma 3. $\square$

Note that the constructed operator $\text{Proj}_{\mathcal{H}_\mathbf{u}^+}(\cdot)$ is provably *idempotent* [1].

Lemma 6. *Operator $\text{Proj}_{\mathcal{H}_\mathbf{u}^+}(\cdot)$ is idempotent.*

Proof. Given PD precision matrix $\mathbf{P}$, operator $\text{Proj}_{\mathcal{H}_\mathbf{u}^+}(\cdot)$ performs the aforementioned algorithm on $\mathbf{C} = \mathbf{P}^{-1}$ so that the output $\hat{\mathbf{C}}$ can be written as a linear combination of $\{\mathbf{U}_i\}_{i=1}^K$ and $\{\mathbf{V}_i\}_{i=K+1}^N$, *i.e.*,

$$\hat{\mathbf{C}} = \sum_{i=1}^{K} \mu_i \mathbf{U}_i + \sum_{i=K+1}^{N} \mu_i \mathbf{V}_i. \tag{3.54}$$

By construction, $\{\mathbf{U}_i\}_{i=1}^K$ and $\{\mathbf{V}_i\}_{i=K+1}^N$ are mutually orthogonal. Also by construction, $\mu_1 = \langle \mathbf{C}, \mathbf{U}_1 \rangle$, $\mu_t = \min(\langle \mathbf{E}_{t-1}, \mathbf{U}_t \rangle, \mu_{t-1})$ for $t \in \{2, \ldots, K\}$, and $\mu_t = \min(\langle \mathbf{E}_{t-1}, \mathbf{V}_t \rangle, \mu_{t-1})$ for $t \in \{K+1, \ldots, N\}$. Thus, when applying the algorithm again to output $\hat{\mathbf{C}}$, new coefficient μ_1' is

$$\mu_1' = \langle \hat{\mathbf{C}}, \mathbf{U}_1 \rangle \overset{(a)}{=} \mu_1 \tag{3.55}$$

where (a) is true due to (3.54) and by orthogonality of $\{\mathbf{U}_i\}_{i=1}^{K}$ and $\{\mathbf{V}_i\}_{i=K+1}^{N}$. Each new coefficient μ_t' for $t \in \{2, \ldots, K\}$ is

$$\mu_t' = \min(\langle \mathbf{E}_{t-1}, \mathbf{U}_t \rangle, \mu_{t-1}) \tag{3.56}$$

$$\overset{(a)}{=} \min(\langle \hat{\mathbf{C}} - \sum_{i=1}^{t-1} \mu_i \mathbf{U}_i, \mathbf{U}_t \rangle, \mu_{t-1}) \tag{3.57}$$

$$\overset{(b)}{=} \min(\langle \hat{\mathbf{C}}, \mathbf{U}_t \rangle, \mu_{t-1}) \tag{3.58}$$

$$\overset{(c)}{=} \min(\mu_t, \mu_{t-1}) \overset{(d)}{=} \mu_t \tag{3.59}$$

where (a) is true due to derivation for $\mathbf{E}_{t-1}$ in (3.33), (b) is true by orthogonality of $\{\mathbf{U}_i\}_{i=1}^{K}$, (c) is true due to derivation for $\hat{\mathbf{C}}$ in (3.54) and by orthogonality of $\{\mathbf{U}_i\}_{i=1}^{K}$ and $\{\mathbf{V}_i\}_{i=K+1}^{N}$, and (d) is true by definition of μ_t. A similar proof can show also that each new coefficient μ_t' for $t \in \{K+1, \ldots, N\}$ equals to μ_t. Thus, $\hat{\mathbf{C}}$ is left unchanged. We conclude that $\mathrm{Proj}_{\mathcal{H}_{\mathbf{u}}^+}(\mathrm{Proj}_{\mathcal{H}_{\mathbf{u}}^+}(\mathbf{P})) = \mathrm{Proj}_{\mathcal{H}_{\mathbf{u}}^+}(\mathbf{P})$. $\square$

The flow diagram of the proposed algorithm is shown in Figure 3.1, where it shows the summary of stages that are needed to approximate $\mathbf{C}$'s projection $\mathbf{C}_{\mathbf{\Omega}^\perp}$ to $\mathbf{\Omega}^\perp$ as $\hat{\mathbf{C}}_{\mathbf{\Omega}^\perp}$.

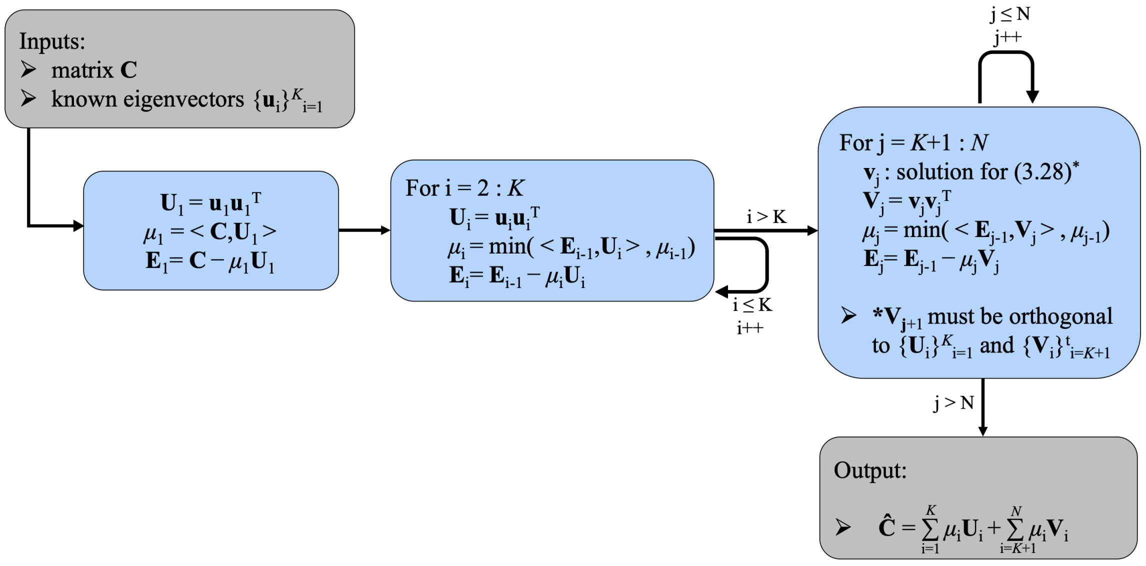

Inputs:
matrix $\mathbf{C}$
known eigenvectors $\{\mathbf{u}_i\}_{i=1}^{K}$
$\mathbf{U}_1 = \mathbf{u}_1\mathbf{u}_1^T$
$\mu_1 = \langle \mathbf{C}, \mathbf{U}_1 \rangle$
$\mathbf{E}_1 = \mathbf{C} - \mu_1\mathbf{U}_1$
For $i = 2 : K$
$\mathbf{U}_i = \mathbf{u}_i\mathbf{u}_i^T$
$\mu_i = \min(\langle \mathbf{E}_{i-1}, \mathbf{U}_i \rangle, \mu_{i-1})$
$\mathbf{E}_i = \mathbf{E}_{i-1} - \mu_i\mathbf{U}_i$
$i > K$
$i \leq K$
$i{+}{+}$
$j \leq N$
$j{+}{+}$
For $j = K+1 : N$
$\mathbf{v}_j$: solution for (3.28)*
$\mathbf{V}_j = \mathbf{v}_j\mathbf{v}_j^T$
$\mu_j = \min(\langle \mathbf{E}_{j-1}, \mathbf{V}_j \rangle, \mu_{j-1})$
$\mathbf{E}_j = \mathbf{E}_{j-1} - \mu_j\mathbf{V}_j$
*$\mathbf{V}_{j+1}$ must be orthogonal to $\{\mathbf{U}_i\}_{i=1}^{K}$ and $\{\mathbf{V}_i\}_{i=K+1}^{t}$
$j > N$
Output:
$\hat{\mathbf{C}} = \sum_{i=1}^{K} \mu_i\mathbf{U}_i + \sum_{i=K+1}^{N} \mu_i\mathbf{V}_i$

Figure 3.1: Steps of our algorithm to construct the approximation of $\mathbf{C}$'s projection into $\mathcal{H}_{\mathbf{u}}^{+}$.

Chapter 4

Graph Laplacian Matrix Estimation

In this chapter, the proposed algorithm to estimate a sparse graph Laplacian from an empirical covariance matrix with a prior on the first K eigenvectors is discussed in detail. This algorithm is a graph learning method, which combines the formulation from graphical Lasso [8] and the projection operator developed in Chapter 3. More specifically, the algorithm in this chapter applies the projection operator from Chapter 3 after each iteration of coordinate descent (BCD) in GLASSO, to compute the most suitable graph Laplacian matrix $\mathbf{L}^* \in \mathcal{H}_{\mathbf{u}}^+$ given input empirical covariance matrix $\bar{\mathbf{C}}$.

First, in Section 4.1, the optimization problem inspired by graphical Lasso is formulated. Next, in Section 4.2, the hybrid algorithm to solve the optimization problem is proposed.

4.1 Optimization Problem

Given an empirical covariance matrix $\bar{\mathbf{C}}$ computed from data, a variant of GLASSO [96] is posed to estimate a PD sparse graph Laplacian matrix $\mathbf{L}$ as follows:

$$\min_{\mathbf{L} \in \mathcal{H}_{\mathbf{u}}^{+}} \quad \mathrm{Tr}(\mathbf{L}\bar{\mathbf{C}}) - \log\det \mathbf{L} + \rho \, \|\mathbf{L}\|_1 \tag{4.1}$$

where $\rho > 0$ is a shrinkage parameter for the ℓ_1 norm. The only difference from the original GLASSO formulation in (2.28) is that (4.1) has an additional constraint $\mathbf{L} \in \mathcal{H}_{\mathbf{u}}^{+}$. Because $\mathcal{H}_{\mathbf{u}}^{+}$ is a convex set and the objective is convex, (4.1) is a convex optimization problem.

4.2 Hybrid Graphical Lasso / projection Algorithm

(4.1) can be solved iteratively using the developed projection operator in Chapter 3 and a variant of the *block Coordinate descent* (BCD) algorithm in [85]. In a nutshell, BCD minimizes the objective by optimizing one row and column of a desired solution matrix at a time. Specifically, first the *dual* of GLASSO can be written as follows. Note first that the ℓ_1 norm in (4.1) can be written as [29]:

$$\|\mathbf{L}\|_1 = \max_{\|\mathbf{U}\|_\infty \leq 1} \quad \mathrm{Tr}(\mathbf{L}\mathbf{U}) \tag{4.2}$$

where $\|\mathbf{U}\|_\infty$ is the maximum absolute value element of the symmetric matrix $\mathbf{U}$. Then, the dual problem of GLASSO that solves for an estimated covriance matrix

$\mathbf{C} = \mathbf{L}^{-1}$ is

$$\min_{\mathbf{C}^{-1} \in \mathcal{H}^+} \quad -\log\det \mathbf{C}, \quad \text{s.t.} \ \ \|\mathbf{C} - \bar{\mathbf{C}}\|_\infty \leq \rho \tag{4.3}$$

where $\mathbf{C} = \bar{\mathbf{C}} + \mathbf{U}$ implies that the primal and dual variables are related via $\mathbf{L} = (\bar{\mathbf{C}} + \mathbf{U})^{-1}$ [29]. To solve (4.3), one row-column pair in $\mathbf{C}$ is updated in (4.3) in each iteration. Specifically, first the rows / columns of $\mathbf{C}$ are reordered so that the optimizing row / column are swapped to the last. Then $\bar{\mathbf{C}}$ and $\mathbf{C}$ can be partitioned into blocks:

$$\mathbf{C} = \begin{pmatrix} \mathbf{C}_{11} & \mathbf{c}_{12} \\ \mathbf{c}_{12}^\top & c_{22} \end{pmatrix}, \quad \bar{\mathbf{C}} = \begin{pmatrix} \bar{\mathbf{C}}_{11} & \bar{\mathbf{c}}_{12} \\ \bar{\mathbf{c}}_{12}^\top & \bar{c}_{22} \end{pmatrix}. \tag{4.4}$$

[29] showed that the optimal $\mathbf{c}_{12}$ is the solution to the following linearly constrained quadratic programming problem:

$$\mathbf{c}_{12} = \arg\min_{\mathbf{y}} \{\mathbf{y}^\top \mathbf{C}_{11}^{-1} \mathbf{y}\}, \ \ \text{s.t.} \ \ \|\mathbf{y} - \bar{\mathbf{c}}_{12}\|_\infty \leq \rho. \tag{4.5}$$

The objective is convex in (4.5) since $\mathbf{C}_{11}^{-1}$ is PSD based on Theorem 3 in [29].

The algorithm to solve (4.1) is thus as follows. Instead of the primal problem (4.1), its corresponding dual (4.3) is solved instead—iteratively updating one row / column of $\mathbf{C}$ using (4.5). (4.5) is a box-constrained quadratic program which is solved using an interior point procedure in [29]. Specifically, they showed that solving (4.5) is equivalent to solving the dual problem [8]:

$$\min_{\boldsymbol{\beta}} \left\{ \frac{1}{2} \|\mathbf{C}_{11}^{1/2} \boldsymbol{\beta} - b\|^2 + \rho \|\boldsymbol{\beta}\|_1 \right\} \tag{4.6}$$

where $b = \mathbf{C}_{11}^{-1/2}\bar{\mathbf{c}}_{12}$. If $\boldsymbol{\beta}$ is the solution to (4.6), then $\mathbf{c}_{12} = \mathbf{C}_{11}\boldsymbol{\beta}$ is the solution to (4.5). (4.6) is a lasso regression problem that can be easily solved.

Each time one row / column of $\mathbf{C}$ is updated, $\mathbf{L} = \mathbf{C}^{-1}$ is projected to convex cone $\mathcal{H}_{\mathbf{u}}^{+}$ using the projection operator described in Chapter 3. The procedure of iteratively updating row / column and projecting to $\mathcal{H}_{\mathbf{u}}^{+}$ is repeated till convergence. Note that *both steps are computed using estimated covariance* $\mathbf{C}$ *directly*, and thus inversion to graph Laplacian $\mathbf{L} = \mathbf{C}^{-1}$ is not necessary until convergence, when a solution is computed. The worst-case complexity of our algorithm is the same as the computation time of GLASSO with BCD in [8] which is $\mathcal{O}(N^3)$ where N is a number of rows / columns of data matrix.

Algorithm 1 shows the pseudo-code to estimate a symmetric matrix $\mathbf{C}$ from an empirical covariance matrix $\bar{\mathbf{C}}$ computed from data. Denote by $\mathbf{C}_{jj}$ a sub-matrix of $\mathbf{C}$ by removing row j and column j. Denote by $\bar{\mathbf{c}}_j$ column j of $\bar{\mathbf{C}}$ with the diagonal element $\bar{\mathbf{C}}(j,j)$ removed.

Algorithm 1: GLASSO/Projection

```
1. Initialize: $\mathbf{C}^{(0)} = \bar{\mathbf{C}} + \rho\mathbf{I}$.
2. Repeat the following steps till convergence (by cycling around the columns):
    (a) Rearrange the rows/columns so that the target column is last.
    (b) Let $\mathbf{C}^{(j-1)}$ denote the current iterate. Solve the dual problem in (4.5):
        $\hat{\mathbf{y}} := \arg\min_{\mathbf{y}} \{\mathbf{y}^\top(\mathbf{C}_{jj}^{(j-1)})^{-1}\mathbf{y} \text{ s.t. } \|\mathbf{y} - \bar{\mathbf{c}}_j\|_\infty \le \rho\}$
    (c) Update the row/column (off-diagonal) of the covariance: $\mathbf{C}^{(j)}$ is $\mathbf{C}^{(j-1)}$
        with row/column $\mathbf{C}^{(j)}$ replaced by $\hat{\mathbf{y}}$.
    (d) Project $\mathbf{C}$ to the subspace with $K$ desired last eigenvectors $\mathcal{H}_{\mathbf{u}}^{+}$.
3. Check the convergence condition. If convergence is reached, the final result is
   $\mathbf{L} = \mathbf{C}^{-1}$.
```

Chapter 5

Experiments

In this chapter, results of conducted experiments to test the proposed algorithm's performance are presented. In Section 5.1, experimental setup for synthetic datasets is first described. In Section 5.2, results assuming the first K eigenvectors are known are presented. Finally, in Section 5.3, results using eigenvectors computed using the fast graph Fourier transform (FGFT) method [30] are provided.

5.1 Experiments Setups for Synthetic Datasets

Experiments were conducted using synthetic datasets to evaluate the performance of the proposed method (Proj-Lasso) and of competing schemes: Graphical Lasso (GLASSO) [8], graph Laplacian learning with Gaussian probabilistic prior (GL-SigRep) [9] and diagonally dominant generalized graph Laplacian estimation under structural and Laplacian constraints (DDGL) [6]. The convergence tolerance for these algorithms was set to $\epsilon = 10^{-4}$, and the regularization parameter was $\rho = e^{-6}$.

To simulate ground truth graphs, 20 nodes were first randomly located in 2D space, and the Erdos-Renyi model [97] was used to determine their connectivity with probability 0.6. Erdos-Renyi graph, $\mathcal{G}_{ER}^{(n,p)}$, is a graph with n nodes, where each node is connected to another with probability p. Edge weights were computed using a Gaussian kernel, *i.e.*, $w_{ij} = \exp\left(-d(i,j)^2/2\sigma^2\right)$, where $d(i,j)$ is the Euclidean distance between i and j, and σ was set to 0.5. Edge weights smaller than 0.75 were removed for sparsity. To introduce negative weights, the sign of each edge was flipped with probability 0.5. The generalized graph Laplacian $\mathcal{L} = \mathbf{D} - \mathbf{W} + \text{diag}(\mathbf{W})$ was computed. To generate data from $\mathcal{L}$, covariance matrix $\mathcal{K} = (\mathcal{L} + \epsilon\mathbf{I})^{-1}$ for $\epsilon = 0.5$ was computed. Then dataset $\mathcal{X} = \{\mathbf{x}_i\}_{i=1}^{20}$ was generated from multivariate normal distribution $\mathbf{x} \sim \mathcal{N}(0, \mathcal{K})$. An empirical covariance matrix $\bar{\mathbf{C}}$ was then computed from $\mathcal{X}$ and used as input to different graph learning algorithms.

Three popular graph similarity metrics were employed to evaluate graph Laplacian matrices learned: relative error (RE), DeltaCon and λ-distance [98–101]. The first metric is RE, which computes the relative *Frobenius norm* of the error between the ground truth Laplacian matrix $\mathbf{L}^o$ and the learned matrix $\hat{\mathbf{L}}$. Specifically, RE uses the following equation:

$$\text{RE}(\hat{\mathbf{L}}, \mathbf{L}^o) = \frac{\|\hat{\mathbf{L}} - \mathbf{L}^o\|_F}{\|\mathbf{L}^o\|_F}. \tag{5.1}$$

DeltaCon similarity function compares the similarities between all node pairs in the two graphs. It satisfies the general properties of other distances, while it also satisfies some properties related to the impact of specific changes [98]. For example, changes resulting in disconnectedness of graphs are penalized more in this function, or changes are more important in sparse graphs than in dense graphs with equal size. Moreover, in the case of weighted graphs, removing the edges with bigger weights has greater impact on the DeltaCon function.

The last metric is λ-distance, which computes the eigenvalues' distance between two matrices as follows:

$$d_\lambda(\mathcal{G}_1, \mathcal{G}_2) = \sqrt{\sum_{i=1}^{k} (\lambda_{1i} - \lambda_{2i})^2} \tag{5.2}$$

where $\{\lambda_{1i}\}_{i=1}^{|\mathcal{V}_1|}$ and $\{\lambda_{2i}\}_{i=1}^{|\mathcal{V}_2|}$ are the eigenvalues of the matrices that represent $\mathcal{G}_1$ and $\mathcal{G}_2$, and k is $\max(|\mathcal{V}_1|, |\mathcal{V}_2|)$ [100, 101].

5.2 Results assuming the First K Eigenvectors

Table 5.1 shows the graph learning performance of different methods evaluated using the aforementioned three metrics.

Metric	GLASSO	GL-SigRep	DDGL	Proj-Lasso with different K's		
				1	2	3
RE	0.9306	0.7740	0.7543	0.7295	0.5191	0.4005
DeltaCon	0.9422	0.9449	0.9407	0.9502	0.9495	0.9586
λ-distance	1e+03	8.7168	28.7918	9.0983	3.3768	2.7125

Table 5.1: Average RE, Deltacon and λ-distance for graph produced with 20 vertices and 20 signals on each node.

The proposed Proj-Lasso was executed using different K's—the number of known eigenvectors corresponding to the smallest K eigenvalues—which in the experiments were the first K eigenvectors of the ground truth matrix $\mathcal{L}$. As shown in Table 5.1, using RE as metric, Proj-Lasso outperformed GLASSO, DDGL and GL-SigRep when $K = 1$, and became even better as k increased. Using λ-distance as metric, Proj-Lasso was always better than its competitors. Using DeltaCon as metric, all the methods are comparable. One can observe that Proj-Lasso's performance evaluated using RE and λ-distance improved significantly as K increased, while the results of DeltaCon metric were not sensitive to K.

Visual comparisons of learned Laplacian matrices are shown in Fig. 5.1. One can observe that Proj-Lasso had better performance than GLASSO and DDGL, *i.e.*, Proj-Lasso was visually closer to the ground truth matrix. GL-SigRep has a visible diagonal pattern, but the off-diagonal terms are overly uniform compared to ground truth. Numerical results show Proj-Lasso is closer to ground truth than GL-SigRep.

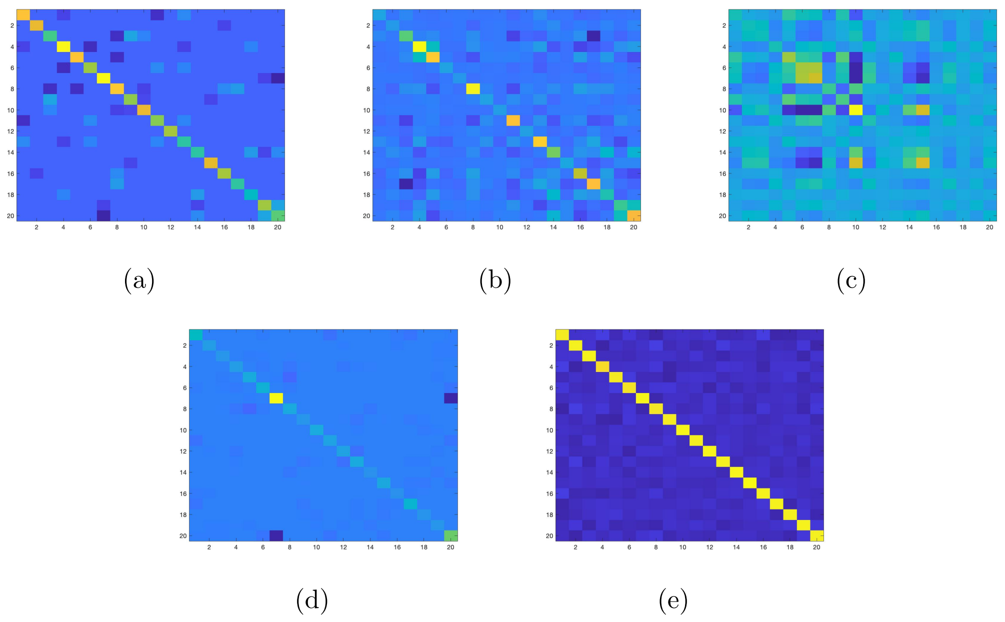


Figure 5.1: **(a)** Ground Truth Laplacian $\mathcal{L}$ with 20 nodes, **(b)** Proposed Proj-Lasso with $K = 1$, **(c)** GLASSO [8], **(d)** DDGL [6] and **(e)** GL-SigRep [9] .

In an another experiment, the size of graph was increased to 30 nodes with the same setup. The visual results for the proposed method and the competing methods are shown in Figure 5.2. It can be observed that the result from Proj-Lasso is visually closer to the ground truth than GLASSO and DDGL. This is particularly obvious for the diagonal entries. Again, Proj-Lasso is closer to the ground truth than GL-SigRep in numerical comparisons.

For the last experiment, in Figure 5.3 the results of different methods are shown via

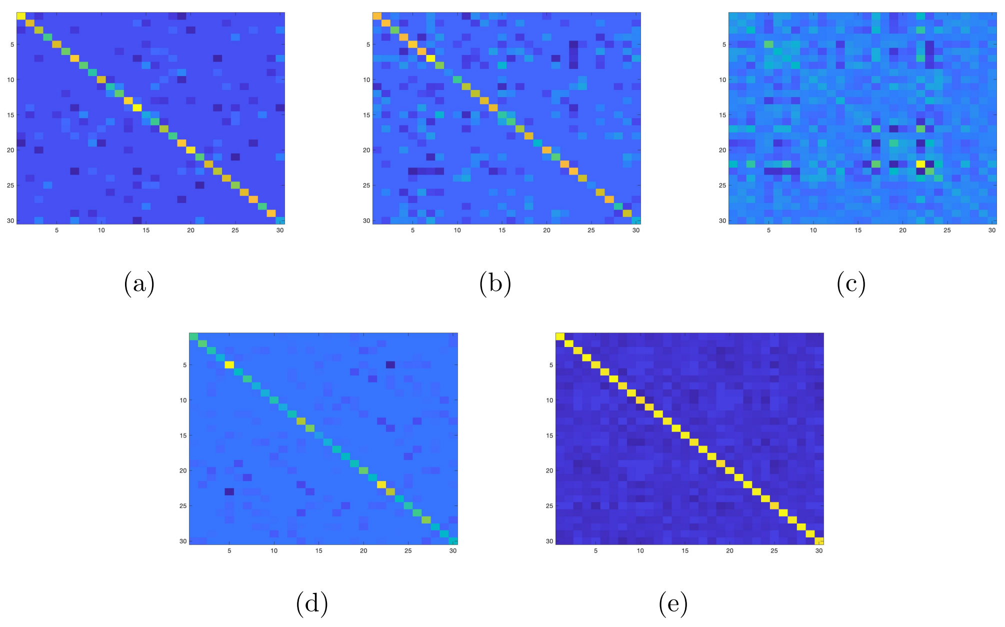


Figure 5.2: **(a)** Ground Truth Laplacian $\mathcal{L}$ with 30 nodes, **(b)** Proposed Proj-Lasso with $K = 1$ (RE $= 0.5318$), **(c)** GLASSO [8] (RE $= 0.9077$), **(d)** DDGL [6] (RE $= 0.5525$) and **(e)** GL-SigRep [9] (RE $= 0.6731$).

graph embedding: each node was assigned a 2D coordinate so that the graph topology can be visualized. The same experimental setup was used as earlier experiments, except edge weights less than 0.3 were removed in this case. While the results from GLASSO (b) and GL-SigRep (e) contain all the edges from the ground truth graph, they also include many additional edges not present in the ground truth graph and thus lack sparsity. In contrast, the result from DDGL (d) is too sparse and does not include all of the edges found in the ground truth graph. The result from Proj-Lasso (c), however, maintains sparsity without compromising accurate edge representation.

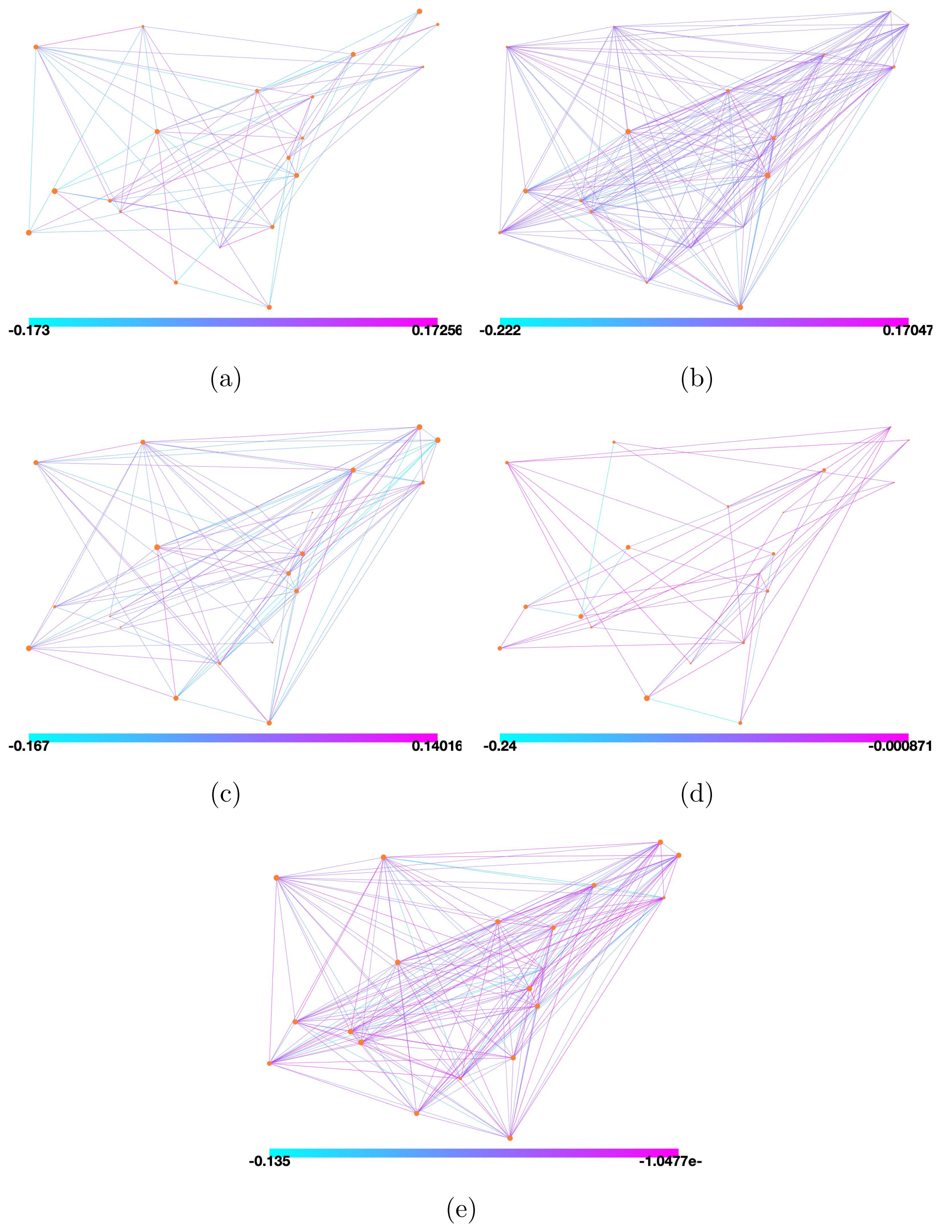


Figure 5.3: **(a)** Ground Truth Laplacian $\mathcal{L}$ with 20 nodes using Erdos-Renyi model with $p = 0.6$ and Gaussian kernel model with $\sigma = 0.5$. Edges with weights less than 0.3 are removed and the sign of each edge is negative with probability 0.5., **(b)** visualized result of GLASSO (RE = **0.8686**) **(c)** visualized result of proposed Proj-Lasso with $K = 1$ (RE = **0.5539**), **(d)** visualized result of DDGL (RE = **0.6655**), and **(e)** visualized result of GL-SigRep (RE = **0.9391**). Edges are coloured according to their weight.

5.3 Results using Eigenvectors from Givens Rotation Method

[30] developed a method based on Givens rotation matrices to approximately diagonalize Laplacian $\mathbf{L}$, *i.e.*,

$$\mathbf{L} \approx \mathbf{S}_1 \cdots \mathbf{S}_J \hat{\mathbf{\Lambda}} \mathbf{S}_J^\top \cdots \mathbf{S}_1^\top \tag{5.3}$$

where $\mathbf{S}_1, \ldots, \mathbf{S}_J$ are Givens rotation matrices that are both sparse and orthogonal, and $\hat{\mathbf{\Lambda}}$ is a near-diagonal matrix. $\mathbf{T} = \mathbf{S}_J^\top \cdots \mathbf{S}_1^\top$ can be interpreted as a fast *Graph Fourier Transform* (FGFT) that approximates the original eigen-matrix of $\mathbf{L}$. $\mathbf{T}$ is sparse since each $\mathbf{S}_j$ is sparse, and thus computation of transform coefficients $\boldsymbol{\alpha} = \mathbf{Tx}$ can be fast. J is a parameter to trade off the complexity of the transform $\mathbf{T}$ and the GFT approximation error.

Metric	$K = 1$	$K = 2$	$K = 3$	$K = N$
RE	1.3375	1.3585	1.3595	1.4224
DeltaCon	0.9295	0.9237	0.9323	0.9283
λ-distance	2.3943	2.4010	2.4010	2.5390

Table 5.2: Average relative errors of Proj-Lasso using different number of first K eigenvectors from Givens rotation method [30] as inputs.

In this experiment, given an estimated covariance matrix, first, sparse K eigenvectors were computed using the FGFT method, which were columns of the multiplications of Givens rotation matrices (*i.e.*, the first K rows of $\mathbf{T}$). These orthogonal vectors were used as the first K eigenvector prior, and the remaining $N-K$ eigenvectors were estimated using the proposed algorithm. Parameter J for Givens methods was set to $J = 2000$. Note that in this case, the first K orthogonal vectors were not the actual first K eigenvectors of the target inverse covariance matrix, but that they were the preferred vectors for faster computation.

Table 5.2 shows the performance of Proj-Lasso using different numbers of rows from $\mathbf{T}$ as prior eigenvectors ($K = 1, \ldots 3$, and $K = N$). This table shows the tradeoff between potential speedup in computing transform coefficients using sparse eigenvectors and inverse covariance matrix estimation accuracy. One can observe that Proj-Lasso has smaller errors for both metrics RE and λ-distance when K is smaller *i.e.*, when fewer computed eigenvectors from the FGFT method are used.

Chapter 6

Conclusion

Computing a most suitable graph structure from available data observations via an efficient graph learning algorithm represents a critical challenge in data science. The emerging field of GSP provides a unique perspective for graph learning via signal representations on the learned graph. Consequently, novel graph learning methods inspired by GSP are capable of robustly estimating complex graph structures, from small and possibly noisy datasets, suitable for real-world data such as those found in brain and social network analysis. In this thesis, a novel GSP-based graph learning algorithm (Proj-Lasso) is proposed.

Given observable signals to compute an empirical covariance matrix $\bar{\mathbf{C}}$, a new graph learning method is proposed to estimate the most likely graph Laplacian matrix $\mathbf{L}$, where the assumption is that the first K eigenvectors of $\mathbf{L}$ are pre-selected based on domain knowledge or an alternative criterion. First, it was proven that the subspace $\mathcal{H}_{\mathbf{u}}^{+}$ of symmetric positive semi-definite (PSD) matrices sharing the first K eigenvectors $\{\mathbf{u}_k\}_{k=1}^{K}$ in a defined Hilbert space is a convex cone. Then, an operator, inspired by the Gram-Schmidt procedure, is constructed to project a positive definite (PD) matrix $\mathbf{P}$ into $\mathcal{H}_{\mathbf{u}}^{+}$. Finally, an efficient algorithm to compute the most probable

$\mathbf{L} \in \mathcal{H}_{\mathbf{u}}^{+}$ given $\bar{\mathbf{C}}$, combining block coordinate descent (BCD) in GLASSO and the proposed projection operator, is designed.

Experimental results show that the proposed algorithm outperformed competing graph learning schemes when the first K eigenvectors are known. Proj-Lasso does not require full matrix-decomposition—which is resource-intensive—to construct a projection operator. Additionally, optimizing the covariance matrix directly—instead of using its inverse—in the estimation of a graph Laplacian results in further computational benefits in implementation. To the best of our knowledge, the proposed algorithm is the first in the literature that employs assumptions about *eigenvectors* of graph Laplacian $\mathbf{L}$.

Future work is possible in a number of different directions. First, the implementation of the current algorithm can be made more computation-efficient and memory-efficient. More precisely, these improvements can be achieved by leveraging fast sparse precision matrix estimation algorithms such as CLIME [102] instead of GLASSO. Further, the optimization problem used in the projection operator can be solved faster using alternating projections inspired by projection on convex sets (POCS) [103]. Experimentally, real-world datasets from practical applications should be used to generate new experimental results. Image processing (*e.g.*, image and transform coding), social networks (*e.g.*, networks containing observations about voting behaviour), and neurological imaging (*e.g.*, fMRI data) are candidate applications that can provide interesting real-world datasets.

Bibliography

[1] M. Vetterli, J. Kovavcević, and V. Goyal, *Foundations of Signal Processing.* Cambridge University Press, 2014.

[2] W. Hu, G. Cheung, and A. Ortega, “Intra-prediction and generalized graph Fourier transform for image coding,” in *IEEE Signal Processing Letters*, vol. 22, no.11, pp. 1913–1917, November 2015.

[3] A. Ortega, *Introduction to Graph Signal Processing.* Cambridge University Press, 2021.

[4] M. Rodriguez, J. Leskovec, D. Balduzzi, and B. Schölkopf, “Uncovering the structure and temporal dynamics of information propagation,” *Network Science*, vol. 2, pp. 26–65, 04 2014.

[5] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, “Learning graphs from data: A signal representation perspective,” *IEEE Signal Processing Magazine*, vol. 36, p. 44–63, May 2019.

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

[7] S. Kumar, J. Ying, J. V. de Miranda Cardoso, and D. Palomar, "Structured graph learning via laplacian spectral constraints," in *Advances in Neural Information Processing Systems* (H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, eds.), vol. 32, Curran Associates, Inc., 2019.

[8] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics (Oxford, England)*, vol. 9, pp. 432–41, 08 2008.

[9] 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, 2016.

[10] 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, p. 83–98, May 2013.

[11] W. Hu, G. Cheung, A. Ortega, and O. Au, "Multi-resolution graph Fourier transform for compression of piecewise smooth images," in *IEEE Transactions on Image Processing*, vol. 24, no.1, pp. 419–433, January 2015.

[12] J. Zeng, G. Cheung, Y. Chao, I. Blanes, J. Serra-Sagristà, and A. Ortega, "Hyperspectral image coding using graph wavelets," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 1672–1676, 2017.

[13] X. Su, M. Rizkallah, T. Maugey, and C. Guillemot, "Graph-based light fields representation and coding using geometry information," in *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 4023–4027, 2017.

[14] W. Hu, X. Li, G. Cheung, and O. Au, "Depth map denoising using graph-based transform and group sparsity," in *IEEE International Workshop on Multimedia Signal Processing*, (Pula, Italy), October 2013.

[15] J. Pang, G. Cheung, A. Ortega, and O. C. Au, "Optimal graph Laplacian regularization for natural image denoising," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, (Brisbane, Australia), April 2015.

[16] J. Pang and G. Cheung, "Graph laplacian regularization for image denoising: Analysis in the continuous domain," *IEEE Transactions on Image Processing*, vol. 26, p. 1770–1785, Apr 2017.

[17] W.-t. Su, G. Cheung, R. Wildes, and C.-W. Lin, "Graph neural net using analytical graph filters and topology optimization for image denoising," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 8464–8468, 2020.

[18] Y. Bai, G. Cheung, X. Liu, and W. Gao, "Graph-based blind image deblurring from a single photograph," in *IEEE Transactions on Image Processing*, vol. 28, no.3, pp. 1404–1418, March 2019.

[19] G. Cheung, E. Magli, Y. Tanaka, and M. Ng, "Graph spectral image processing," in *Proceedings of the IEEE*, vol. 106, no.5, pp. 907–930, May 2018.

[20] C. Dinesh, G. Cheung, and I. V. Bajić, "Point cloud denoising via feature graph Laplacian regularization," *IEEE Transactions on Image Processing*, vol. 29, pp. 4143–4158, 2020.

[21] F. Wang, Y. Wang, G. Cheung, and C. Yang, "Graph sampling for matrix completion using recurrent Gershgorin disc shift," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2814–2829, 2020.

[22] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. 01 2009.

[23] O. Banerjee and L. Ghaoui, "Model selection through sparse max likelihood estimation model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data," *Journal of Machine Learning Research*, vol. 9, 08 2007.

[24] M. Gomez Rodriguez, J. Leskovec, and A. Krause, "Inferring networks of diffusion and influence," *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining - KDD '10*, 2010.

[25] C. Groendyke, D. Welch, and D. Hunter, "Bayesian inference for contact networks given epidemic data," *Scandinavian Journal of Statistics*, vol. 38, pp. 600–616, Sept. 2011.

[26] F. Graham, F. Chung, A. M. Society, and C. B. of the Mathematical Sciences, *Spectral Graph Theory*. Regional conference series in mathematics, Conference Board of the mathematical sciences, 1996.

[27] E. Pavez, H. Egilmez, Y. Wang, and A. Ortega, "GTT: Graph template transforms with applications to image coding," in *31st Picture Coding Symposium*, (Cairns, Australia), May 2015.

[28] K. J. Friston, "Functional and effective connectivity in neuroimaging: A synthesis," *Human Brain Mapping*, vol. 2, no. 1-2, pp. 56–78, 1994.

[29] O. Banerjee and L. Ghaoui, "Model selection through sparse max likelihood estimation," *Journal of Machine Learning Research*, vol. 9, 08 2007.

[30] L. Le Magoarou, N. Tremblay, and R. Gribonval, "Analyzing the approximation error of the fast graph Fourier transform," in *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pp. 45–49, 2017.

[31] W. Hoffmann, "Iterative algorithms for Gram-Schmidt orthogonalization," *Computing*, vol. 41, no. 4, pp. 335–348, 1989.

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

[33] G. Strang, "The discrete cosine transform," *SIAM Review*, vol. 41, no. 1, pp. 135–147, 1999.

[34] 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*, vol. 5, no. 2, pp. 360–374, 2019.

[35] C. Yang, G. Cheung, and W. Hu, "Graph metric learning via gershgorin disc alignment," in *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5530–5534, 2020.

[36] C. Tomasi and R. Manduchi, "Bilateral filtering for gray and color images," in *Proceedings of the IEEE International Conference on Computer Vision*, (Bombay, India), 1998.

[37] A. Gadde, S. K. Narang, and A. Ortega, "Bilateral filter: Graph spectral interpretation and extensions," in *IEEE International Conference on Image Processing*, (Melbourne, Australia), September 2013.

[38] S. Z. Li, "Markov random field models in computer vision," in *Computer Vision — ECCV '94* (J.-O. Eklundh, ed.), (Berlin, Heidelberg), pp. 361–370, Springer Berlin Heidelberg, 1994.

[39] H. Rue and L. Held, *Gaussian Markov Random Fields: Theory And Applications (Monographs on Statistics and Applied Probability)*. Chapman Hall/CRC, 2005.

[40] V. Goyal, "Theoretical foundations of transform coding," *IEEE Signal Processing Magazine*, vol. 18, no. 5, pp. 9–21, 2001.

[41] W. B. Pennebaker and J. L. Mitchell, *JPEG Still Image Data Compression Standard*. New York: Van Nostrand Reinhold, 1992.

[42] G. J. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 22, no.12, December 2012.

[43] H. Malvar, A. Hallapuro, M. Karczewicz, and L. Kerofsky, "Low-complexity transform and quantization in h.264/avc," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, pp. 598 – 603, 07 2003.

[44] G. Shen, W.-S. Kim, S. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," in *IEEE Picture Coding Symposium*, (Nagoya, Japan), December 2010.

[45] J. Han, A. Saxena, V. Melkote, and K. Rose, "Jointly optimized spatial prediction and block transform for video and image coding," *IEEE Transactions on Image Processing*, vol. 21, no. 4, pp. 1874–1884, 2012.

[46] D. A. Huffman, "A method for the construction of minimum-redundancy codes," *Proceedings of the IRE*, vol. 40, no. 9, pp. 1098–1101, 1952.

[47] A. K. Jain, *Fundamentals of Digital Image Processing.* USA: Prentice-Hall, Inc., 1989.

[48] H. Hotelling, "Analysis of a complex of statistical variables into principal components.," *Journal of Educational Psychology*, vol. 24, pp. 498–520, 1933.

[49] N. Meinshausen and P. Buhlmann, "High-dimensional graphs and variable selection with the lasso," *Annals of Statistics*, vol. 34, pp. 1436–1462, 2006.

[50] C.-J. Hsieh, M. Sustik, I. Dhillon, and P. Ravikumar, "Sparse inverse covariance matrix estimation using quadratic approximation," *Advances in Neural Information Processing Systems*, vol. 24, 06 2013.

[51] D. Heckerman, *A Tutorial on Learning with Bayesian Networks*, pp. 33–82. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[52] A. P. Dempster, "Covariance selection," *Biometrics*, vol. 28, no. 1, pp. 157–175, 1972.

[53] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society (Series B)*, vol. 58, pp. 267–288, 1996.

[54] P. Ravikumar, M. J. Wainwright, G. Raskutti, and B. Yu, "High-dimensional covariance estimation by minimizing 1 -penalized log-determinant divergence," *Electron. J. Statist.*, vol. 5, pp. 935–980, 2011.

[55] T. Cai, W. Liu, and X. Luo, "A constrained 1 minimization approach to sparse precision matrix estimation," *Journal of the American Statistical Association*, vol. 106, no. 494, pp. 594–607, 2011.

[56] M. Slawski and M. Hein, "Estimation of positive definite m-matrices and structure learning for attractive gaussian markov random fields," *Linear Algebra and its Applications*, vol. 473, 04 2014.

[57] G. Poole and T. Boullion, "A survey on m-matrices," *SIAM Review*, vol. 16, no. 4, pp. 419–427, 1974.

[58] B. M. Lake and J. B. Tenenbaum, "Discovering structure by learning sparse graph," in *Proceedings of the 33rd Annual Cognitive Science Conference*, 2010.

[59] X. Zhu, Z. Ghahramani, and J. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the Twentieth International Conference on International Conference on Machine Learning*, ICML'03, p. 912–919, AAAI Press, 2003.

[60] R. Mazumder and T. Hastie, "Exact covariance thresholding into connected components for large-scale graphical lasso," *Journal of Machine Learning Research*, vol. 13, 08 2011.

[61] C.-J. Hsieh, M. Sustik, I. Dhillon, P. Ravikumar, and R. Poldrack, "Big and quic: Sparse inverse covariance estimation for a million variable," *Proc. Adv. Neural Inform. Process. Syst.*, pp. 3165–3173, 01 2013.

[62] M. Grechkin, M. Fazel, D. Witten, and S.-I. Lee, "Pathway graphical lasso," *Proceedings of the AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, vol. 2015, pp. 2617–2623, 2015.

[63] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu, "Network tomography: Recent developments," *Statist. Sci.*, vol. 19, pp. 499–517, 08 2004.

[64] S. Ratnasamy and S. McCanne, "Inference of multicast routing trees and bottleneck bandwidths using end-to-end measurements," in *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, vol. 1, pp. 353–360 vol.1, 1999.

[65] M. G. Rabbat, M. J. Coates, and R. D. Nowak, "Multiple-source internet tomography," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 12, pp. 2221–2234, 2006.

[66] S. A. Myers and J. Leskovec, "On the convexity of latent social network inference," in *NIPS*, 2010.

[67] N. Du, L. Song, M. Yuan, and A. Smola, “Learning networks of heterogeneous influence,” in *Advances in Neural Information Processing Systems* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), vol. 25, Curran Associates, Inc., 2012.

[68] R. Pastor-Satorras, C. Castellano, P. Van Mieghem, and A. Vespignani, “Epidemic processes in complex networks,” *Reviews of Modern Physics*, vol. 87, p. 925–979, Aug 2015.

[69] W. Hu, G. Cheung, X. Li, and O. Au, “Graph-based joint denoising and super-resolution of generalized piecewise smooth images,” in *IEEE International Conference on Image Processing*, (Paris, France), October 2014.

[70] X. Liu, D. Zhai, D. Zhao, G. Zhai, and W. Gao, “Progressive image denoising through hybrid graph laplacian regularization: A unified framework,” in *IEEE Transactions on Image Processing*, vol. 23, no.4, pp. 1491–1503, January 2013.

[71] W. Hu, G. Cheung, and M. Kazui, “Graph-based dequantization of block-compressed piecewise smooth images,” in *IEEE Signal Processing Letters*, vol. 23, no.2, pp. 242–246, February 2016.

[72] X. Liu, G. Cheung, X. Ji, D. Zhao, and W. Gao, “Graph-based joint dequantization and contrast enhancement of poorly lit JPEG images,” in *IEEE Transactions on Image Processing*, vol. 28, no.3, pp. 1205–1219, March 2019.

[73] W. Huang, T. A. W. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, “A graph signal processing perspective on functional brain imaging,” *Proceedings of the IEEE*, vol. 106, no. 5, pp. 868–885, 2018.

[74] I. Jabłoński, “Graph signal processing in applications to sensor networks, smart grids, and smart cities,” *IEEE Sensors Journal*, vol. 17, no. 23, pp. 7659–7666, 2017.

[75] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero, “Learning sparse graphs under smoothness prior,” *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Mar 2017.

[76] 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, 2018.

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

[78] 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, 2017.

[79] D. Zhai, X. Liu, D. Zhao, H. Chang, and W. Gao, “Progressive image restoration through hybrid graph laplacian regularization,” in *Data Compression Conference*, (Snowbird, UT), March 2013.

[80] S. I. Daitch, J. Kelner, and D. Spielman, “Fitting a graph to vector data,” in *ICML ’09*, 2009.

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

[82] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1798–1828, 2013.

[83] C. Hu, L. Cheng, J. Sepulcre, K. Johnson, G. Fakhri, Y. Lu, and Q. Li, “A spectral graph regression model for learning brain connectivity of alzheimer’s disease,” *PLOS ONE*, vol. 10, p. e0128136, 05 2015.

[84] V. Kalofolias, “How to learn a graph from smooth signals,” in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics* (A. Gretton and C. C. Robert, eds.), vol. 51 of *Proceedings of Machine Learning Research*, (Cadiz, Spain), pp. 920–929, PMLR, 09–11 May 2016.

[85] S. Wright, “Coordinate descent algorithms,” *Math. Program.*, vol. 151, no. 1, pp. 3–34, 2015.

[86] F. R. K. Chung, *Spectral Graph Theory.* American Mathematical Society, 1997.

[87] D. A. Spielman and S.-H. Teng, “Spectral sparsification of graphs,” *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011.

[88] M. Sundin, A. Venkitaraman, M. Jansson, and S. Chatterjee, “A connectedness constraint for learning sparse graphs,” in *2017 25th European Signal Processing Conference (EUSIPCO)*, pp. 151–155, 2017.

[89] S. Hassan-Moghaddam, N. K. Dhingra, and M. R. Jovanović, "Topology identification of undirected consensus networks via sparse inverse covariance estimation," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, pp. 4624–4629, 2016.

[90] S. Boyd and L. Vandenberghe, *Convex Optimization.* Cambridge University Press, 2004.

[91] G. Golub and C. F. V. Loan, *Matrix Computations (Johns Hopkins Studies in the Mathematical Sciences).* Johns Hopkins University Press, 2012.

[92] R. Horn, R. Horn, and C. Johnson, *Matrix Analysis.* Cambridge University Press, 1990.

[93] D. Dereniowski and M. Kubale, "Cholesky factorization of matrices in parallel and ranking of graphs," in *Parallel Processing and Applied Mathematics*, (Berlin, Heidelberg), pp. 985–992, Springer Berlin Heidelberg, 2004.

[94] A. Knyazev, "Toward the optimal preconditioned eigensolver: Locally optimal block preconditioned conjugate gradient method," *SIAM journal on scientific computing*, vol. 23, no. 2, pp. 517–541, 2001.

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

[96] R. Mazumder and T. Hastie, "The graphical lasso: New insights and alternatives," *Electron. J. Statist.*, vol. 6, pp. 2125–2149, 2012.

[97] P. Erdös and A. Rényi, "On random graphs i," *Publicationes Mathematicae Debrecen*, vol. 6, p. 290, 1959.

[98] D. Koutra, J. Vogelstein, and C. Faloutsos, "DELTACON: A principled massive-graph similarity function," *CoRR*, vol. abs/1304.4657, 2013.

[99] M. Tantardini, F. Ieva, L. Tajoli, and C. Piccardi, "Comparing methods for comparing networks," *Scientific Reports*, vol. 9, 12 2019.

[100] H. Bunke, P. Dickinson, M. Kraetzl, and W. Wallis, *A Graph-Theoretic Approach to Enterprise Network Dynamics.* Progress in Computer Science and Applied Logic, Birkhäuser Boston, 2006.

[101] R. Wilson and P. Zhu, "A study of graph spectra for comparing graphs and trees," *Pattern Recognition*, vol. 41, pp. 2833–2841, 09 2008.

[102] T. Cai, W. Liu, and X. Luo, "A constrained ℓ_1 minimization approach to sparse precision matrix estimation," in *Journal of the American Statistical Association*, vol. 106, pp. 594–607, 2011.

[103] A. Zakhor, "Iterative procedures for reduction of blocking effects in transform image coding," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 2, no. 1, pp. 91–95, 1992.