Learning and Signal Processing over High-Dimensional Graphs

By

SONGYANG ZHANG

DISSERTATION

Submitted in partial satisfaction of the requirements for the degree of

DOCTOR OF PHILOSOPHY

in

Electrical and Computer Engineering

in the

OFFICE OF GRADUATE STUDIES

of the

UNIVERSITY OF CALIFORNIA

DAVIS

Approved:

Zhi Ding, Chair

Shuguang Cui

Lifeng Lai

Committee in Charge

2021

Abstract

Learning and Signal Processing over High-Dimensional Graphs

Learning and signal processing methods over graphs have recently attracted significant attentions in dealing with structured data. Normal (traditional) graphs, however, only capture pairwise relationships among nodes and are not effective in representing and capturing some high-order relationships of data samples. Such high order data interactions are often important in many applications such as Internet of Things (IoT), multimedia processing and network analysis, thereby motivating the exploration of learning and signal processing through high-dimensional graphs. In this dissertation, we investigate theoretical foundations and practical applications of two different high-dimensional graphs: 1) multilayer networks, and 2) hypergraphs. Inspired by the properties of high-dimensional graphs, we also revisit certain aspects of signal processing and learning under normal graphs.

First, we study the behavior analysis of propagation over multilayer networks. Specifically, we focus on the cascading failure over multilayer complex systems. We first propose a scalable tensor-based framework to represent interdependent multilayer networks, before applying this framework to analyze the failure propagation based on a susceptible-infectious-susceptible (SIS) epidemic model. We derive the transition equations and failure threshold to characterize the failure propagation. To make the failure indicator analytically tractable and computationally efficient, we derive its upper and lower bounds, as well as its approximated expressions in special cases.

Second, we investigate signal processing over hypergraphs. Representing hypergraphs as tensors, we proposed a novel framework of hypergraph signal processing (HGSP). Defining a specific form of hypergraph signals and hypergraph signal shifting, we provide an alternative definition of hypergraph Fourier space based on the tensor decomposition, together with the corresponding hypergraph Fourier transform. To better interpret the hypergraph Fourier space, we analyze the resulting hypergraph frequency properties, including the concepts of frequency and bandlimited signals. We also establish theoretical foundation for the HGSP sampling theory and filter designs. Furthermore, we examine its applications in multimedia processing, including three-dimensional (3D) point clouds, images and videos.

Lastly, revisiting the traditional graphs, we investigate the development of graph convolutional networks from the perspective of graph signal processing (GSP). We reexamine the graph spectral convolution in GSP and define conditions for approximating spectrum wavelet via propagation in the vertex domain. We then propose alternative propagation models for the GCN layers and develop a Taylor-series based graph convolutional networks (TGCN) based on the aforementioned approximation conditions. Our experimental results in citation networks and point clouds validate the effectiveness of the proposed TGCN.

Acknowledgement

I would like to express my sincere gratitude to all those who helped me during the writing of this dissertation.

My deepest gratitude goes first and foremost to my advisors, Prof. Zhi Ding and Prof. Shuguang Cui, for offering me the opportunities to work on the cutting-edge research in the advanced signal processing. Their insightful inspirations and kind encouragement motivate me to go over problems and keep moving forward. I am truly grateful for their help in selecting research directions, discussing technical details, revising manuscripts and providing financial supports. Their wisdom benefits me not only in my research, but also in my daily life. From them, I learned how to balance my life and research, as well as how to pursue a righteous life energetically. They provide great help and care in my future career.

I would like to express my heartfelt thanks to my family members, who stand beside me all the time. Thank my parents, Lin Wu and Hongrong Zhang, for their endless love and selfless support. Thank my wife, Jiani Zhou, for her believing and understanding, even though we live separately in a long distance for these years. Their encouragement cheers me up when I am depressed; and their acknowledgement pushes me forward when I achieve success. I could not stand here without the support from them.

I would like to thank all my friends and colleagues at University of California, Davis, especially, Han Zhang, Fuwei Li, Xiaoxiao Wang, Xiaochuan Ma, Zhelun Zhang, Qinwen Deng, Qilian Yu, Carlos Feres, Siyu Qi, Minhui Huang, Taha Bouchoucha, and Lahiru Chamain. Thank Dr. Hang Li at Shenzhen Research Institute of Big Data for his continuous suggestions and help in both my life and research. I also express my gratitude to Yue Xu, Zhi Wang, and Chen Qiu at Beijing University of Posts and Telecommunications for their kindness. Besides, I would show my sincere gratitude to my friends, Longfei Chen and Haiqiang Wang, in China.

Finally, I would like to thank all my committee members, Prof. Lifeng Lai, Prof. Bernard

Levy, and Prof. Xiaodong Li, for their time and support on my qualifying exam, defense and the dissertation. Their valuable comments benefit me a lot in improving my works.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview of Applying Graphs in Engineering

Studies of graphs have received significant attentions over the past decades in many areas, including engineering, biology, and social science. The ability of describing the relationship between different points or entities make graphs important tools in real applications [1]. For example, data, like social networks, biological data, mobility and traffic patterns, resides on complex structures that do not lend themselves to standard tools [2]. Graphs provide ability to model such data and the complex relationship between them. Fig. 1.1 shows several examples of modeling realistic datasets by graphs. Modeling each data point as a node and the correlations as edges, graph-based learning and signal processing methods achieved significant successes in data analysis.

Generally, the research of graphs in engineering can be summarized as the follows:

- *Network sciences:* This area addresses issues such as uncovering community relations, perceived alliances, quantifying connectedness, or determining the relevance of specific agents [3–5]. Researches in this category mainly focus on the structure rather than components in the system. It covers, for example, giant components, clustering coefficients, centralities and path length [6, 7]. It can used in analyzing the structure of the

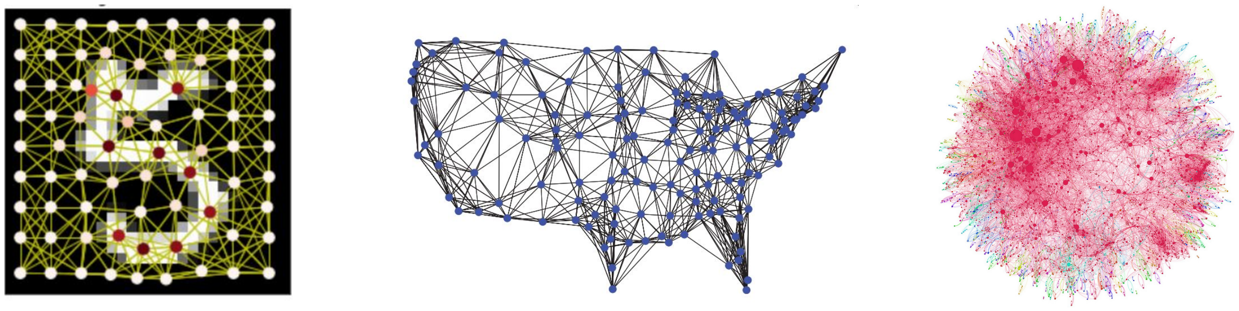

(a) Image [12] (b) Sensor network [13] (c) Citation network [14]

Figure 1.1: Graph Models of Different Datasets.

system and providing instructions for designing a robust system.

- *Network process:* The aim is to model and analyze the propagation over networks, including diffusion of information, epidemic and disease infections, social influence and propagation of failures [8]. Graphs are intuitive models in such problems. Classic models are similar to those in network science. Each entity is modeled as one node and the connections are modeled as edges. Then, a function is attached to each edge to describe the propagation between two nodes. This area is important in real world. A typical example is cascading failure, where a robust structure can be designed to reduce the loss of failure in a system by analyzing the failure propagation [2].

- *Graphical models:* This area focuses on the inference and learning from large datasets [9–11]. The data is modeled as a set of random variables described by a family of Gibbs probability distributions, and the underlying graph captures statistical dependence and conditional independence among the data. Research in this area exploit the distribution defined on the the graph to model and learn the properties of the data.

- *Graph signal processing:* Signal processing defined over graphs aims to develop a similar framework as digital signal processing to analyze the signal in the graph Fourier space [13]. Given a graph, data points are represented by graph signals corresponding to vertices and their internal relationships are represented by the edges. Then, a graph

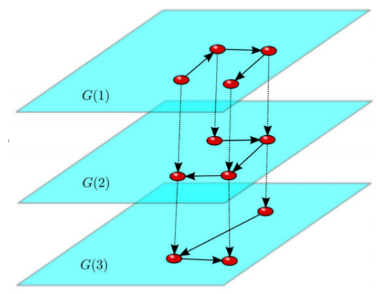
Fourier space is defined based on the eigen-space of the representing matrix (Laplacian or adjacency) to complete signal processing and data analysis tasks, like denoising [15], filter banks [16] and compression [17].

1.2 Motivations of Extending Graphs to High-Dimensional Domains

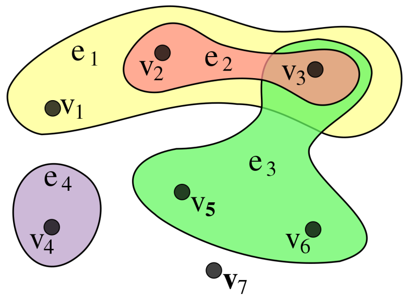
With the coming of big data era, the system and data structure become more and more complex. Although graphs have achieved great successes in dealing with multiple engineering problems, they still exhibit some limitations. Since each edge in normal graphs only connects two nodes, it can only represent pairwise relationships. In general, multi-way correspondences are more informative and precise in signal processing and data analysis [18]. For example, in biology, a disease is usually triggered by multiple genes [19]. The relations between these genes and disease cannot be easily modeled in pairwise as a normal graph. In addition, normal graph is hard to process multiple different relationships. Data, like social network, usually has more than one type of relationship between two nodes. For example, a single person may serve different roles in different relationships, such as a father in family and a teacher in school [20]. There may be several different types of links among all the nodes. How to distinguish such different natures in different levels leverages a problem. All these limitations motivate us to extend the traditional data analysis and signal processing based on normal graphs to high-dimensional domains.

In this dissertation, we investigate applications of two high-dimensional graphs in engineering: 1) multilayer networks, and 2) hypergraphs.

- *Multilayer network* consists of several layers with different natures. An example of three-layer network is shown in Fig. 1.2(a). Multilayer network can be used in representing complex systems with several components and describing relationship with different natures. A typical example is smart grid, which has two major components:



(a) Example of three-layer network



(b) Example of a hypergraph

Figure 1.2: Example of High Dimensional Graphs

the power grid and the communication network. The power stations in the power grid provide energy for the communication network, and the base stations in the communication network jointly control the power stations [22]. Such system can be modeled as a two-layer network, where one layer represents the communication network and the other one represents power grid. Applications, like cascading failure and system stability, can be further analyzed based on the constructed multilayer network.

- *Hypergraph* is composed of nodes and hyperedges connecting more than one nodes [23]. An example is shown in Fig. 1.2(b). The normal graph can be viewed as a special case of hypergraphs in which each hyperedge degrades to connecting exactly two nodes. Hypergraphs have shown great successes in replacing normal graphs in many applications, such as clustering [24], classification [25] and prediction [26]. Hypergraph is a natural alternative of normal graph in the signal processing and data analysis of high dimensional interactions.

More specifically, we focus on the signal processing and behavior analysis over the these two high-dimensional graphs. First, we propose a tensor-based spectral analysis of cascading failure in the multilayer networks. Second, we investigate the developments of hypergraph signal processing (HGSP), as well as its applications in multimedia. Finally, back to normal graphs, we propose a graph convolutional networks based on Taylor expansions.

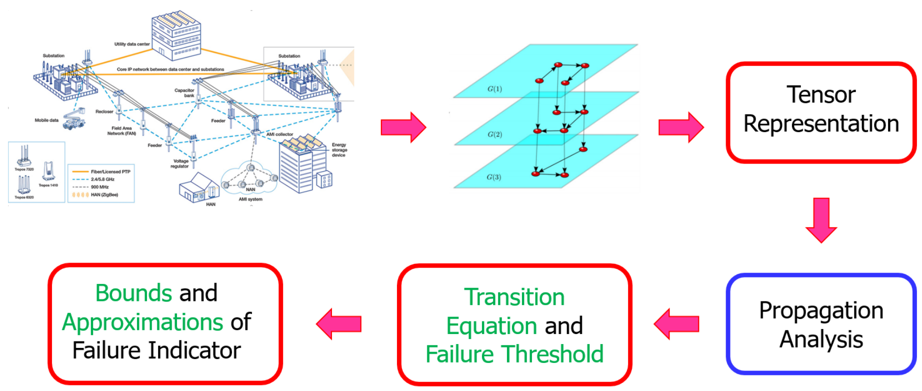

Tensor Representation
Propagation Analysis
Transition Equation and Failure Threshold
Bounds and Approximations of Failure Indicator

Figure 1.3: Diagram of Behavior Analysis of Cascading Failure in Multilayer Networks.

1.3 Overview of Research Topics

The main body of the dissertation includes three parts:

- In the first part, we represent multilayer networks by tensors and investigate the failure propagation under the susceptible-infectious-susceptible (SIS) model shown as Fig. 1.3. We first propose a scalable tensor-based framework to model the physical multilayer complex systems. Then, we analyze the SIS epidemic behavior and derive the transition equation to describe the failure propagation. We show that the spectral radius of transition tensor is a failure indicator with an explicit failure threshold to measure the system reliability. To explore how this indicator depends on the network structure and epidemic parameters, we derive its upper and lower bounds. In addition, we derive the approximations of the failure indicator for the cases where either intra-layer or inter-layer propagation dominates. The analytical results are shown to exceed the performance of some benchmark approximation methods

- In the second part, we propose a new framework of hypergraph signal processing (HGSP) based on tensor representation to generalize the traditional graph signal processing (GSP) to tackle high-order interactions. We introduce the core concepts of HGSP and define the hypergraph Fourier space. We then study the spectrum properties of hypergraph Fourier transform and explain its connection to mainstream digital

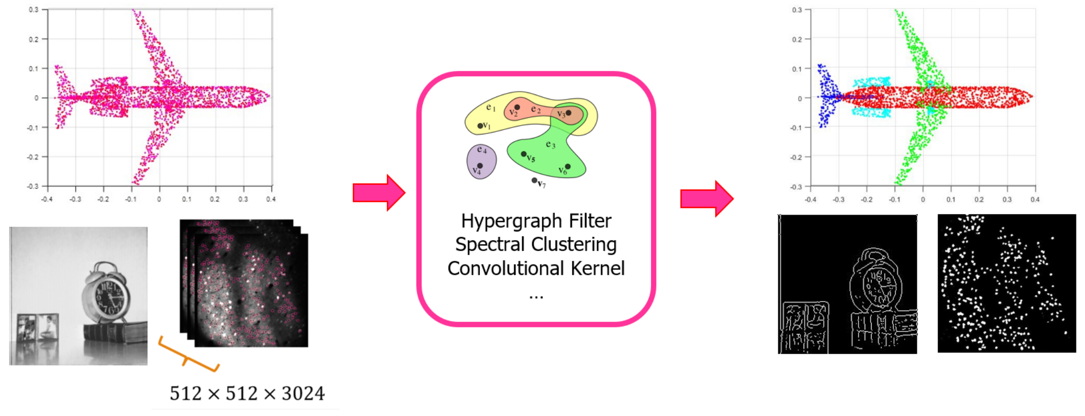


Figure 1.4: Examples of HGSP-based Applications in Multimedia Datasets.

signal processing. We derive the novel hypergraph sampling theory and present the fundamentals of hypergraph filter design based on the tensor framework. We present HGSP-based methods for several signal processing and data analysis applications. In addition, we investigate the applications of HGSP in multimedia, including images, videos and point clouds. Our experimental results demonstrate significant performance improvement using our HGSP framework over traditional signal processing solutions.

- Finally, we revisit the fundamentals of graph wavelet and explores the utility of signal propagation in the vertex domain to approximate the spectral wavelet-kernels. We first derive the conditions for representing the graph wavelet-kernels via vertex propagation. We next propose alternative propagation models for GCN layers based on Taylor expansions. We further analyze the choices of detailed graph representations for TGCNs. Experiments on citation networks, multimedia datasets and synthetic graphs demonstrate the advantage of Taylor-based GCN (TGCN) in the node classification problems over the traditional GCN methods.

Chapter 2

Introduction of Tensor and Tensor Operations

Due to the limitations of matrix in representing high-order data structures, high dimensional algebra tools, especially tensor, has been introduced in dealing with high dimensional issues. In this chapter, we briefly introduce some tensor basics used in this dissertation.

2.1 Introduction of Tensor

Generally speaking, tensors can be interpreted as multi-dimensional arrays. The order of a tensor is the number of indices needed to label a component of that array [27]. For example, a third-order tensor has three indices. In fact, scalars, vectors and matrices are all special cases of tensors: a scalar is a zeroth-order tensor; a vector is a first-order tensor; a matrix is a second-order tensor; and an M-dimensional array is an Mth-order tensor [28]. Generalizing a 2-D matrix, we represent the entry at the position $(i_1, i_2, \cdots, i_M)$ of an Mth-order tensor $\mathbf{T} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_M}$ by $t_{i_1 i_2 \cdots i_M}$ in the rest of the dissertation.

Below are some useful definitions and operations of tensor related to this dissertation.

2.1.1 Symmetric and Diagonal Tensors

- A tensor is *super-symmetric* if its entries are invariant under any permutation of their indices [29]. For example, a third-order $\mathbf{T} \in \mathbb{R}^{I \times I \times I}$ is super-symmetric if its entries t_{ijk}'s satisfy

$$t_{ijk} = t_{jik} = t_{kij} = t_{kji} = t_{jik} = t_{jki} \quad i, j, k = 1, \cdots, I. \tag{2.1}$$

  Analysis of super-symmetric tensors, which is shown to be bijectively related to homogeneous polynomials, could be found in [30, 31].

- A tensor $\mathbf{T} \in \mathbb{R}^{I_1 \times I_2 \cdots \times I_N}$ is *super-diagonal* if its entries $t_{i_1 i_2 \cdots i_N} \neq 0$ only if $i_1 = i_2 = \cdots = i_N$. For example, a third-order $\mathbf{T} \in \mathbb{R}^{I \times I \times I}$ is super-diagonal if its entries $t_{ijk} \neq 0$ only exists for $i = j = k$,while all other entries are zero.

2.2 Tensor Operations

Tensor analysis is developed based on tensor operations [32–34].

- The *tensor outer product* between an Pth-order tensor $\mathbf{U} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_P}$ with entries $u_{i_1 \ldots i_P}$ and an Qth-order tensor $\mathbf{V} \in \mathbb{R}^{J_1 \times J_2 \times \ldots \times J_Q}$ with entries $v_{j_1 \ldots j_Q}$ is denoted by $\mathbf{W} = \mathbf{U} \circ \mathbf{V}$. The result $\mathbf{W} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_P \times J_1 \times J_2 \times \ldots \times J_Q}$ is an $(P + Q)$-th order tensor, whose entries are calculated by

$$w_{i_1 \ldots i_P j_1 \ldots j_Q} = u_{i_1 \ldots i_P} \cdot v_{j_1 \ldots j_Q}. \tag{2.2}$$

  The major use of the tensor outer product is to construct a higher order tensor with several lower order tensors. For example, the tensor outer product between vectors

$\mathbf{a} \in \mathbb{R}^M$ and $\mathbf{b} \in \mathbb{R}^N$ is denoted by

$$\mathbf{T} = \mathbf{a} \circ \mathbf{b}, \tag{2.3}$$

where the result $\mathbf{T}$ is a matrix in $\mathbb{R}^{M \times N}$ with entries $t_{ij} = a_i \cdot b_j$ for $i = 1, 2, \cdots, M$ and $j = 1, 2, \cdots, N$. Now, we introduce one more vector $\mathbf{c} \in \mathbb{R}^Q$, where

$$\mathbf{S} = \mathbf{a} \circ \mathbf{b} \circ \mathbf{c} = \mathbf{T} \circ \mathbf{c}. \tag{2.4}$$

Here, the result $\mathbf{S}$ is a third-order tensor with entries $s_{ijk} = a_i \cdot b_j \cdot c_k = t_{ij} \cdot c_k$ for $i = 1, 2, \cdots, M$, $j = 1, 2, \cdots, N$ and $k = 1, 2, \cdots, Q$.

- The *n-mode product* between a tensor $\mathbf{U} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_P}$ and a matrix $\mathbf{V} \in \mathbb{R}^{\mathbf{J} \times \mathbf{I_n}}$ is denoted by $\mathbf{W} = \mathbf{U} \times_{\mathbf{n}} \mathbf{V} \in \mathbb{R}^{I_1 \times I_2 \times \cdots \times I_{n-1} \times J \times I_{n+1} \times \cdots \times I_P}$. Each element in $\mathbf{W}$ is defined as

$$w_{i_1 i_2 \cdots i_{n-1} j i_{n+1} \cdots i_P} = \sum_{i_n=1}^{I_n} u_{i_1 \cdots i_P} v_{j i_n}, \tag{2.5}$$

  where the main function is to adjust the dimension of a specific order. For example, in Eq. (2.5), the dimension of the nth order of $\mathbf{U}$ is changed from I_n to J.

- The *Kronecker product* of matrices $\mathbf{U} \in \mathbb{R}^{I \times J}$ and $\mathbf{V} \in \mathbb{R}^{P \times Q}$ is defined as

$$\mathbf{U} \otimes \mathbf{V} = \begin{bmatrix} u_{11}\mathbf{V} & u_{12}\mathbf{V} & \cdots & u_{1J}\mathbf{V} \\ u_{21}\mathbf{V} & u_{22}\mathbf{V} & \cdots & u_{2J}\mathbf{V} \\ \vdots & \vdots & \ddots & \vdots \\ u_{I1}\mathbf{V} & u_{I2}\mathbf{V} & \cdots & u_{IJ}\mathbf{V} \end{bmatrix} \tag{2.6a}$$

  to generate an $IP \times JQ$ matrix.

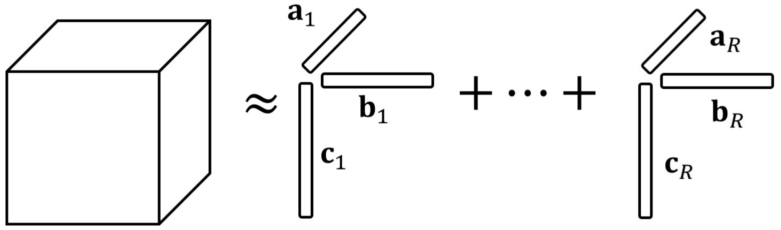


Figure 2.1: CP Decomposition of a Third-order Tensor.

- The *Khatri-Rao product* between $\mathbf{U} \in \mathbb{R}^{I\times K}$ and $\mathbf{V} \in \mathbb{R}^{J\times K}$ is defined as

$$\mathbf{U} \odot \mathbf{V} = [\mathbf{u}_1 \otimes \mathbf{v}_1 \quad \mathbf{u}_2 \otimes \mathbf{v}_2 \quad \cdots \quad \mathbf{u}_K \otimes \mathbf{v}_K]. \tag{2.7}$$

- The *Hadamard product* between $\mathbf{U} \in \mathbb{R}^{P\times Q}$ and $\mathbf{V} \in \mathbb{R}^{P\times Q}$ is defined as

$$\mathbf{U} * \mathbf{V} = \begin{bmatrix} u_{11}v_{11} & u_{12}v_{12} & \cdots & u_{1Q}v_{1Q} \\ u_{21}v_{21} & u_{22}v_{22} & \cdots & u_{2Q}v_{2Q} \\ \vdots & \vdots & \ddots & \vdots \\ u_{P1}v_{P1} & u_{P2}v_{P2} & \cdots & u_{PQ}v_{PQ} \end{bmatrix}. \tag{2.8}$$

2.3 Tensor Decomposition

Similar to the eigen-decomposition for matrix, tensor decomposition analyzes tensors via factorization. The CANDECOMP/PARAFAC (CP) decomposition is a widely used method, which factorizes a tensor into a sum of component rank-one tensors [27,35]. For example, a third order tensor $\mathbf{T} \in \mathbb{R}^{I\times J\times K}$ is decomposed into

$$\mathbf{T} \approx \sum_{r=1}^{R} \mathbf{a}_r \circ \mathbf{b}_r \circ \mathbf{c}_r, \tag{2.9}$$

where $\mathbf{a}_r \in \mathbb{R}^I$, $\mathbf{b}_r \in \mathbb{R}^J$, $\mathbf{c}_r \in \mathbb{R}^K$ and R is a positive integer known as rank, which leads to the smallest number of rank-one tensors in the decomposition. The process of CP decomposition for a third-order tensor is illustrated in Fig. 2.1.

There are several extensions and alternatives of the CP decomposition. For example, the orthogonal-CP decomposition [36] decomposes the tensor using an orthogonal basis. For an M-th order N-dimension tensor $\mathbf{T} \in \mathbb{R}^{\overbrace{N \times N \times ... \times N}^{\text{M times}}}$, it can be decomposed by the orthogonal-CP decomposition as

$$\mathbf{T} \approx \sum_{r=1}^{R} \lambda_r \cdot \mathbf{a}_r^{(1)} \circ ... \circ \mathbf{a}_r^{(M)}, \tag{2.10}$$

where $\lambda_r \geq 0$ and the orthogonal basis is $\mathbf{a}_r^{(i)} \in \mathbb{R}^N$ for $1 \leq i \leq M$. More specifically, the orthogonal-CP decomposition has a similar form to the eigen-decomposition when $M = 2$ and $\mathbf{T}$ is super-symmetric.

2.4 Tensor Spectrum

The eigenvalues and spectral space of tensors are significant topics in tensor algebra. The research of tensor spectrum has achieved great progress in recent years. In particular, Lim and the others developed theories of eigenvalues, eigenvectors, singular values, and singular vectors for tensors based on a constrained variational approach such as the Rayleigh quotient [39]. Qi and the others in [37,38] presented a more complete discussion of tensor eigenvalues by defining two forms of tensor eigenvalues, i.e., the E-eigenvalue and the H-eigenvalue. Chang and the others [29] further extended the work of [37, 38]. Other works including [40, 41] further developed the theory of tensor spectrum.

Chapter 3

Spectral Analysis of Propagation Behavior over Multilayer Networks

3.1 Introduction

Studies of multilayer complex systems have received significant attentions over the past decades in many areas, including engineering, biology, and social science. A multilayer complex system is usually defined as a system consisting of several interdependent components with different natures [1]. A typical example is smart grid, which has two major components: the power grid and the communication network. The power stations in the power grid provide energy for the communication network, and the communication nodes in the communication network jointly control the power stations[22]. Other examples include human brains, social organizations and so on [1]. In the literature, multilayer complex systems are often modeled as multilayer networks where each layer corresponds to one type of components [43].

One of the most important threats in the multilayer complex systems is the cascading failure: the failure of a small fraction of the system triggers failures of other parts [8].

[1]Part of this chapter is reprinted, with permission, from [S. Zhang, H. Zhang, H. Li, and S. Cui, "Tensor-based Spectral Analysis of Cascading Failures over Multilayer Complex Systems," in *2018 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Oct. 2018].

In complex systems, the failure of one component may cause new failures not only in the same layer but also across layers, leading to severe damages to the whole system. For example, on September 28th of 2003, the damage of some power lines by the storms caused several power stations down in Italy. The failure of these power stations led to a failure of the communication network, which in turn caused a further breakdown of power stations, affecting a total of 56 million people in Europe [44]. As such, cascading failures in multilayer complex systems deserve great attentions. A deep understanding over this issue would benefit the future design of a robust and resilient multilayer system.

To analyze the cascading failures over complex systems, one key problem is how to model the failure propagation. Typical methods are based on giant components [8], logical connections [45] and statistical propagation probability. Among them, the epidemic model, which originally describes the spread of virus, is widely used. For example, Wang *et al.* [46] studied the failure propagation based on a susceptible-infected-susceptible (SIS) model in the two-layer networks. Granell *et al.* [47] studied the behavior of susceptible-infected-exposed (SIE) epidemics in multiplex networks where each layer has the same number of nodes. Other works focusing on epidemic and cascading failure like [48] mainly investigated the epidemic spread over random graphs. The behavior of failure propagation can be characterized and analyzed by the probability transition in the epidemic model. However, most of existing literatures only focused on the networks with up to two layers.

Another key problem of analyzing cascading failures in multilayer systems is how to represent multilayer networks mathematically. Almost all traditional methods are matrix-based. For example, Wang *et al.* [46] used two different matrices to represent the intra-layer connections and inter-layer connections, and Domenico *et al.* [49] proposed to represent each layer with an individual adjacency matrix. However, these matrix-based representations have some significant limitations. For example, Wang's method limits the number of layers up to two, and the method in [49] cannot handle the inter-layer propagation. To this end, tensor-based methods have been introduced [50], whose power in representing high dimensional graphs makes it especially attractive in complex network analysis.

In this chapter, we represent multilayer networks by tensors and investigate the failure propagation under the SIS model. We first propose a scalable tensor-based framework to model the physical multilayer complex systems. Then, we analyze the SIS epidemic behavior and derive the transition equation to describe the failure propagation. We show that the spectral radius of transition tensor is a failure indicator with an explicit failure threshold to measure the system reliability. To explore how this indicator depends on the network structure and epidemic parameters, we derive its upper and lower bounds. In addition, we derive the approximations of the failure indicator for the cases where either intra-layer or inter-layer propagation dominates. The analytical results are shown to exceed the performance of some benchmark approximation methods.

The rest of this chapter is organized as follows. In Section 3.2, the tensor representation of multilayer networks is introduced. Then the proposed tensor-based framework for physical multilayer systems is given in Section 3.3. The failure model and mathematical formulations of failure propagation are addressed in Section 3.4. We derive the bounds and approximations for failure indicator under certain conditions in Section 3.5. The numerical results are provided in Section 3.6. Finally, the chapter is concluded in Section 3.7.

3.2 Multilayer Networks in Tensor Representation

In this section, we introduce how we use tensor to describe multilayer networks with the same number of nodes in each layer.

Suppose that a multilayer network $\mathcal{A}$ has M layers with N nodes in each layer. As each layer has the same number of nodes, we may consider these nodes as the projections of the same "objects". That is, by projecting N objects onto M layers, we obtain the multilayer network $\mathcal{A}$ that has M layers with N nodes in each layer. Mathematically, the process of projecting objects can be viewed as a tensor product, and the network connections can be represented by the tensor.

Next, we show how to represent multilayer networks by tensor. We use Greek letters $\hat{\alpha}, \hat{\beta}, ...$ to indicate each layer and latin letters $i, j, ...$ to indicate each object. Given N objects $X = \{x_1, ..., x_N\}$, we can construct a contravariant basis $\mathbf{e}^i$ and a corresponding covariant basis $\mathbf{e}_i$ in $\mathbb{R}^N$ to characterize each object x_i. Similarly, given M layers $L = \{l_1, ..., l_M\}$, we can construct basis $\mathbf{e}^{\hat{\alpha}}$ and $\mathbf{e}_{\hat{\alpha}}$ in $\mathbb{R}^M$ to characterize each layer $l_{\hat{\alpha}}$. Then, the connectivity of objects can be represented by a second-order tensor $\mathbf{E} = \sum_{i,j=1}^{N} a_i^j \mathbf{e}_i \circ \mathbf{e}^j \in \mathbb{R}^{N \times N}$ and the connectivity of layers can be represented by $\mathbf{F} = \sum_{\hat{\alpha},\hat{\beta}=1}^{M} b_{\hat{\alpha}}^{\hat{\beta}} \mathbf{e}_{\hat{\alpha}} \circ \mathbf{e}^{\hat{\beta}} \in \mathbb{R}^{M \times M}$, where a_i^j and $b_{\hat{\alpha}}^{\hat{\beta}}$ are the weights. Following the same way, the connectivity between the projected nodes of the objects in the layers can be represented by a forth-order tensor $\mathbf{A} = \sum_{i,j=1}^{N} \sum_{\hat{\alpha},\hat{\beta}=1}^{M} w_{i,\hat{\alpha}}^{j,\hat{\beta}} \mathbf{e}_{\hat{\alpha}} \circ \mathbf{e}^{\hat{\beta}} \circ \mathbf{e}_i \circ \mathbf{e}^j \in \mathbb{R}^{M \times M \times N \times N}$. More specifically, the weights $w_{i,\hat{\alpha}}^{j,\hat{\beta}}$ are set as one or zero to indicate the existence of connectivity. That is, the element $A_{i,\hat{\alpha}}^{j,\hat{\beta}} = 1$ if a directed link exists from the projected node of x_j in layer $l_{\hat{\beta}}$ to that of x_i in layer $l_{\hat{\alpha}}$; otherwise, $A_{i,\hat{\alpha}}^{j,\hat{\beta}} = 0$.

With the tensor representation above, we are able to describe any multilayer network with the same number of nodes in each layer. However, in practice, some physical complex systems cannot be modeled as such regular multilayer networks directly. In the next section, we will introduce a tensor-based framework to model irregular multilayer cyber-physical systems.

3.3 Tensor-based Framework for Complex System

In this section, we propose a framework to model a cyber-physical system as a multilayer network that can be represented in tensor, and discuss its mathematical properties.

3.3.1 Cyber-Physical System

Consider a cyber-physical system consisting of three subsystems and four types of objects: 1) an auto-driving vehicular network composed of N_1 vehicles (Type-1); 2) a smart grid

Table 3.1: Functions of Objects

Index	Function	Type-1	Type-2	Type-3	Type-4
1	Power Generation			✓	
2	Power Transmission	✓	✓	✓	✓
3	Power Storage	✓	✓	✓	✓
4	Communication	✓	✓	✓	✓
5	Computation	✓	✓	✓	✓
6	Human Operator	✓	✓	✓	✓
7	Driving	✓			

consisting of N_2 sub-stations (Type-2) and N_3 main stations (Type-3); and 3) a cellular network consisting of N_4 base stations (Type-4). The functions of each type of objects are shown in Table 3.1. These four types of objects work together to guarantee the whole system in proper operation. Next, we will show how we construct the multilayer network to model such a cyber-physical system based on the interplay among different functions of these objects.

3.3.2 Construction of the Multilayer Network

We first define seven layers based on the functions of all objects shown in Table 3.1. The indices of layers are the sames as those of the functions. To obtain the nodes in the multilayer network, we project all four types of original objects onto seven layers. Then, we define directed links between nodes as the influence indicators. More specifically, we add a directed edge from node a to node b if node a can influence the status of b. The existence and the direction of links depend on the physical property of the original objects. In this way, we obtain a multilayer network with $M = 7$ layers and $N = \sum_{i=1}^{4} N_i$ nodes in each layer. Note that not all objects have functions in all layers. For example, vehicles do not have the "power generation" function. This issue is handled by isolating the corresponding nodes.

Next, we discuss the network structure in detail. In such a cyber-physical system, objects coordinate their own functions and influence other objects via multiple levels of functions.

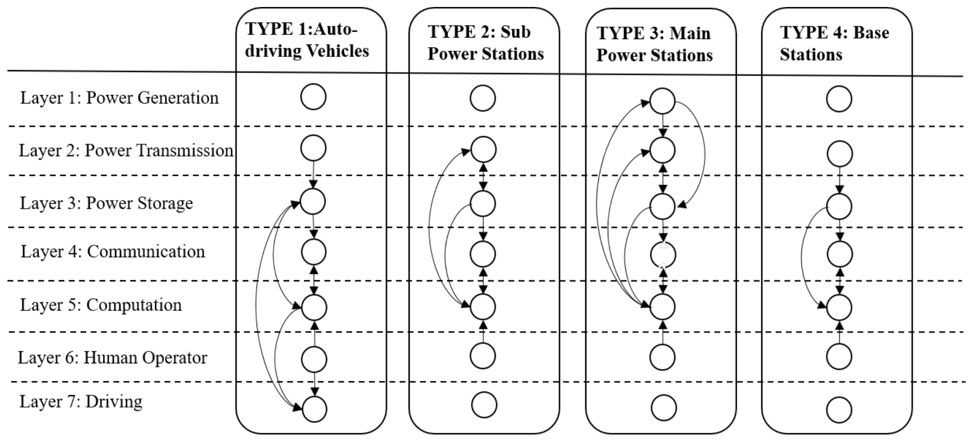

TYPE 1:Auto-driving Vehicles
TYPE 2: Sub Power Stations
TYPE 3: Main Power Stations
TYPE 4: Base Stations
Layer 1: Power Generation
Layer 2: Power Transmission
Layer 3: Power Storage
Layer 4: Communication
Layer 5: Computation
Layer 6: Human Operator
Layer 7: Driving

Figure 3.1: Model of Inter-layer Connections

That is, different functions of the same object or the same functions of different objects may influence each other. Thus, the inter-layer connections exist within the projections of a same object while the intra-layer connections take care of all possible interactions between objects.

For the inter-layer structure, the links represent the direction of failure propagation among the functions of a given object, which deterministically depend on the physical properties of the objects. For different types of objects, the inter-layer links are shown in Fig. 3.1.

- Type-1: A vehicle has six functions except for the power generation function. The power transmitted to the vehicle is stored in the power storage, such that there is a link from layer 2 to layer 3. The communication, computation and driving functions all need energy provided from the power storage. Thus, there are links between layer 3 and layers 4, 5, 7 in both directions. Besides, layers 4 and 5 connect since the computation function needs to exchange information with other vehicles via its communication function. There are also links from layer 6 to layers 5 and 7, since the human drivers can give instructions to the computation function or drive the vehicle manually.

- Type-2: The sub-stations have five functions, and the projected nodes in layers 1 and 7 are isolated. The connections are similar to Type-1 objects among layers $2-6$. The difference is that there are additional links from layers 5, 3 to layer 2, since the computation function in the power stations also controls its load of power transmission and the power transmission can transmit the power stored in the storage. In the sub-stations, human workers can control other functions by giving instructions to the computation functions. Thus, the layer 6 only has connections to layer 5.

- Type-3: The inter-layer structure for a main station is almost the same as that of sub-stations. Note that the main stations have the function of power generation which generates power stored locally or transmitted to other places. Thus, there are additional links from layer 1 to layers 2, 3 and from layer 5 to layer 1.

- Type-4: The base station has the functions of layers $2-6$. The links are similar to those of Type-2 objects.

Now, we discuss the intra-layer structure. In layers 1, 3, 5, 6, 7, the projected nodes do not have intra-layer connections since functions in these layers work only within a same object. In layer 2, the nodes may transmit energy to each other. Since the energy transmission may influence the load balance of nodes, intra-layer links exist and the directions of links are the same as the energy flow. Similar to layer 2, communication nodes in layer 4 exchange information with each other and thus there exist intra-layer connections, whose directions are the same as the flows of information.

3.3.3 Tensor Representation

With the process described above, we obtain a typical multilayer network consisting of M layers with N nodes in each layer. As introduced in Section 3.2, the connections of such a multilayer network can be represented mathematically by the adjacency tensor, denoted by $\mathbf{A} = \{A_{i,\hat{\alpha}}^{j,\hat{\beta}}\} \in \mathbb{R}^{M\times M\times N\times N}$.

To avoid confusion, the intra-layer connections of layer $\hat{\alpha}$ are specified as $\mathbf{A}_{\hat{\alpha}}^{\hat{\alpha}} \in \mathbb{R}^{N\times N}$ with element $A_{i,\hat{\alpha}}^{j,\hat{\alpha}}$ for all i, j. The inter-layer connections from layer $\hat{\beta}$ to $\hat{\alpha}$ are specified as $\mathbf{A}_{\hat{\alpha}}^{\hat{\beta}} \in \mathbb{R}^{N\times N}$ with element $A_{i,\hat{\alpha}}^{j,\hat{\beta}}$ for all i, j. Shown in Fig. 3.1, the same type of objects have the same inter-layer structure. Suppose that object $o(i)$ represents the Type-i object, the inter-layer connections for Type-i objects are specified as $\mathbf{A_{int}}(i) \in \mathbb{R}^{M\times M}$ with element $A_{o(i),\hat{\alpha}}^{o(i),\hat{\beta}}$ for all $\hat{\alpha}$, $\hat{\beta}$.

3.3.4 Properties of the Tensor-based Framework

Identification of Projections

One important property is that the tensor framework preserves the information about whether certain objects has certain types of functions. In particular, an isolated node in one layer implies that an object does not have the corresponding function. This property helps us identify the functions of objects. To identify whether object i has the function in layer $\hat{\alpha}$, we just need to check if the degree $D(i,\hat{\alpha}) = \sum_{(j,\hat{\beta})} A_{i,\hat{\alpha}}^{j,\hat{\beta}} + \sum_{(j,\hat{\beta})} A_{j,\hat{\beta}}^{i,\hat{\alpha}}$ equals zero.

Tensor Flattening

The forth-order tensor $\mathbf{A}$ in $\mathbb{R}^{M\times M\times N\times N}$ can be flattened into second-order tensors in two ways: 1) $\mathbf{F_1} \in \mathbb{R}^{MN\times MN}$ with each element $F_{N(\hat{\alpha}-1)+i}^{N(\hat{\beta}-1)+j} = A_{i,\hat{\alpha}}^{j,\hat{\beta}}$; or 2) $\mathbf{F_2} \in \mathbb{R}^{NM\times NM}$ with each element $F_{M(i-1)+\hat{\alpha}}^{M(j-1)+\hat{\beta}} = A_{i,\hat{\alpha}}^{j,\hat{\beta}}$. These two flattening methods provide two ways to interpret the network structure. In the first method, the flattened multilayer network has M clusters with N nodes in each cluster. The nodes in the same cluster have the same function (belong to the same layer). In the second method, the flattened network has N clusters with M nodes in each cluster. Here, the nodes in the same cluster are from the same original object. The above methods flatten the fourth-order tensor $\mathbf{A}$ into different matrices, which will be helpful in analyzing the behavior of cascading failures in Section 3.5

Diagonal Inter-layer Adjacency Tensor

As we discussed in Section 3.3, the inter-layer connections only exist between the nodes that are from the same object, which means that, for any $\hat{\alpha} \neq \hat{\beta}$, there is $A_{i,\hat{\alpha}}^{j,\hat{\beta}} = 0$ when $i \neq j$;

and $A_{i,\hat{\alpha}}^{j,\hat{\beta}} \in \{0,1\}$, otherwise. Thus, the adjacency tensor $\mathbf{A}_{\hat{\alpha}}^{\hat{\beta}}$ for any pair of layers $\hat{\alpha} \neq \hat{\beta}$ is diagonal. This property helps us simplify the analysis of failure propagation in Section 3.5.

3.4 Spectral Analysis of Cascading Failure

In this section, we analyze the cascading failure for the multilayer complex system in our tensor-based framework. The transition equations and failure indicator are derived to describe the propagation and measure the system stability.

3.4.1 Failure Model

We consider the SIS model in discrete time [51] to model the failure propagation. According to the SIS model, each node has two possible states: susceptible (not fail) or infectious (fail). At each time stamp, the infectious node may transmit disease (cause failure) to other nodes through directed links at certain infection rates, or it may cure itself spontaneously at a self-cure rate. The initial attack make several nodes infectious. This model forms a Markov chain where the probability of a node being infected only depends on the state of the network in the previous time slot.

In our tensor-based framework, the nodes in the same layer correspond to the same function; hence the nodes in the same layer have the same self-cure rate and infection rate. The notations of the epidemic model are given as follows:

- $\mu_{\hat{\alpha}}$: the self-cure rate for nodes in layer $\hat{\alpha}$;
- $\theta_{\hat{\alpha}}^{\hat{\beta}}$: the infection rate describing the failure propagation probability from nodes in layer $\hat{\beta}$ to those in layer $\hat{\alpha}$;
- $P_{i,\hat{\alpha}}(t)$: the failure probability of the projected node of object i in layer $\hat{\alpha}$ at time t;
- $\epsilon_{i,\hat{\alpha}}(t)$: the transition probability that the projected node of object i in layer $\hat{\alpha}$ transits from infectious state to susceptible state at time t;

- $\sigma_{i,\hat{\alpha}}(t)$: the transition probability that the projected node of object i in layer $\hat{\alpha}$ remains susceptible at time t.

To derive the failure probability, we first consider the probability of a node being in the susceptible state. An infectious node becomes susceptible in a given time slot if two events happen simultaneously [52]: 1) it cures itself; and 2) it is not infected by its neighbors. Then,

$$\epsilon_{i,\hat{\alpha}}(t) = \mu_{\hat{\alpha}} \prod_{j,\hat{\beta}} [1 - \theta_{\hat{\alpha}}^{\hat{\beta}} A_{i,\hat{\alpha}}^{j,\hat{\beta}} P_{j,\hat{\beta}}(t)] \tag{3.1}$$

Similarly, a susceptible node remains susceptible when none of its neighbors infect it. Then,

$$\sigma_{i,\hat{\alpha}}(t) = \prod_{j,\hat{\beta}} [1 - \theta_{\hat{\alpha}}^{\hat{\beta}} A_{i,\hat{\alpha}}^{j,\hat{\beta}} P_{j,\hat{\beta}}(t)] \tag{3.2}$$

Based on the state transition described above, the failure probability of the projected node of object i in layer $\hat{\alpha}$ at time t can be calculated recursively by the transition equation as

$$\begin{align} P_{i,\hat{\alpha}}(t) =& 1 - \{\epsilon_{i,\hat{\alpha}}(t-1) P_{i,\hat{\alpha}}(t-1) + \sigma_{i,\hat{\alpha}}(t-1)[1 - P_{i,\hat{\alpha}}(t-1)]\} \tag{3.3} \\ =& 1 - \prod_{j,\hat{\beta}} [1 - T_{i,\hat{\alpha}}^{j,\hat{\beta}} P_{j,\hat{\beta}}(t-1)], \tag{3.4} \end{align}$$

where the coefficients $\{T_{i,\hat{\alpha}}^{j,\hat{\beta}}\}$ are written as

$$T_{i,\hat{\alpha}}^{j,\hat{\beta}} = (1 - \mu_{\hat{\alpha}}) \delta_{i,\hat{\alpha}}^{j,\hat{\beta}} + \theta_{\hat{\alpha}}^{\hat{\beta}} A_{i,\hat{\alpha}}^{j,\hat{\beta}}, \tag{3.5}$$

with $1 \leq i, j \leq N$, $1 \leq \hat{\alpha}, \hat{\beta} \leq M$, and $\delta_{i,\hat{\alpha}}^{j,\hat{\beta}} = 1$ if $(j, \hat{\beta}) = (i, \hat{\alpha})$; otherwise, $\delta_{i,\hat{\alpha}}^{j,\hat{\beta}} = 0$.

We define the transition tensor $\mathbf{T} \in \mathbb{R}^{M \times M \times N \times N}$ with elements $T_{i,\hat{\alpha}}^{j,\hat{\beta}}$. As shown in Eq. (3.4), given the system connections, the failure probability of a certain node only depends on the failure probabilities of all the nodes in the previous time slot and the coefficients

$\{T_{i,\hat{\alpha}}^{j,\hat{\beta}}\}$. If we consider all the nodes together, the transition equations are characterized by the transition tensor $\mathbf{T}$.

3.4.2 Failure Threshold

Now, we analyze the cascading failure based on the transition equations. Considering the steady state of the projected node of object i in layer $\hat{\alpha}$ where we have $P_{i,\hat{\alpha}}(\tau) = P_{i,\hat{\alpha}}(\tau - 1)$, the transition equation is equivalent to

$$\tilde{P}_{i,\hat{\alpha}} = 1 - \prod_{j,\hat{\beta}}[1 - T_{i,\hat{\alpha}}^{j,\hat{\beta}} \tilde{P}_{j,\hat{\beta}}], \tag{3.6}$$

where $\tilde{P}_{i,\hat{\alpha}}$ is the failure probability of the projected node of object i in layer $\hat{\alpha}$ at the steady state. Following [52, 53] and linearize Eq. (3.6), we arrive at a similar conclusion as [54]: when the spectral radius of the $MN \times MN$ flattened transition tensor $\rho(\mathbf{T}) < 1$, $\tilde{P}_{i,\hat{\alpha}} = 0$ exists for the projected node of any object i in any layer $\hat{\alpha}$, where the spectral radius refers to the largest absolute value of the eigenvalues. This conclusion implies that there are no failed nodes at the steady state if $\rho(\mathbf{T}) < 1$. Thus, the spectral radius of the transition tensor $\rho(\mathbf{T})$ is a failure indicator to measure the stability of the system with an explicit failure threshold $\rho(\mathbf{T}) = 1$.

In practice, we can justify the system stability via the spectral radius of the transition tensor in a given multilayer network with known epidemic parameters. However, the complexity of calculating $\rho(\mathbf{T})$ directly could be very high which is polynomial w.r.t the size of matrix [55]. Since the size of the transition tensor is usually very large in real applications, such direct calculation is not cost-efficient. In addition, there is no closed-form expression of the spectral radius for the cases with arbitrary layer structures and epidemic parameters. To tackle these difficulties, we manage to derive the bounds and approximations under certain assumptions to make the failure indicator analytically tractable and computationally efficient.

3.5 Approximation of Spectral Radius of Transition Tensor

In this section, we derive the upper and lower bounds for the spectral radius of the transition tensor. Simplified results for some special cases are also given.

3.5.1 Preliminary Results

We start with some properties of spectral radius.

Lemma 3.1. *If the adjacency tensor* $\mathbf{A} \in \mathbb{R}^{M \times M \times N \times N}$ *of a network can be divided into* K *adjacency tensors* $\mathbf{A_i} \in \mathbb{R}^{M \times M \times N \times N}$*, where* $\mathbf{A} = \mathbf{A_1} + ... + \mathbf{A_K}$*, we have*

$$\rho(\mathbf{A}) = \rho(\mathbf{A_1} + ... + \mathbf{A_K}) \geq \max_{1 \leq i \leq K} \{\rho(\mathbf{A_i})\}. \tag{3.7}$$

The proof of *Lemma 3.1* is in [46]. It shows that the spectral radius of a network is larger than the largest spectral radius of its sub-networks.

Lemma 3.2. *For a Hermitian diagonal block matrix* $\mathbf{A}$ *represented by* $\mathbf{A} = diag([\mathbf{B_1} \quad ... \quad \mathbf{B_K}])$*, its spectral radius can be calculated by*

$$\rho(\mathbf{A}) = \max_{1 \leq i \leq K} \{\rho(\mathbf{B_i})\}. \tag{3.8}$$

The proof of *Lemma 3.2* is in Appendix A.1. It shows that, for a Hermitian diagonal block matrix, its spectral radius is the same as the largest spectral radius of the blocks in its diagonal.

Besides, for the multilayer networks in which there are only inter-layer connections within the projections of the same object, we have the following lemma.

Lemma 3.3. *Given the weight matrix* $\mathbf{W} \in \mathbb{R}^{M\times M}$ *of inter-layer connections, denoted by*

$$\mathbf{W} = \begin{bmatrix} 0 & w_{\hat{2}}^{\hat{1}} & \cdots & w_{\hat{M}}^{\hat{1}} \\ w_{\hat{1}}^{\hat{2}} & 0 & \cdots & w_{\hat{M}}^{\hat{2}} \\ \vdots & \vdots & \ddots & \vdots \\ w_{\hat{1}}^{\hat{M}} & w_{\hat{2}}^{\hat{M}} & \cdots & 0 \end{bmatrix}, \tag{3.9}$$

and the identity matrix $\mathbf{I_N} \in \mathbb{R}^{N\times N}$, *the inter-layer adjacency matrix* $\mathbf{S} \in \mathbb{R}^{MN\times MN}$ *is given as*

$$\mathbf{S} = \begin{bmatrix} 0 & w_{\hat{2}}^{\hat{1}}\mathbf{I_N} & \cdots & w_{\hat{M}}^{\hat{1}}\mathbf{I_N} \\ w_{\hat{1}}^{\hat{2}}\mathbf{I_N} & 0 & \cdots & w_{\hat{M}}^{\hat{2}}\mathbf{I_N} \\ \vdots & \vdots & \ddots & \vdots \\ w_{\hat{1}}^{\hat{M}}\mathbf{I_N} & w_{\hat{2}}^{\hat{M}}\mathbf{I_N} & \cdots & 0 \end{bmatrix}. \tag{3.10}$$

In this way, all the eigenvalues of $\mathbf{S}$ *are the eigenvalues of* $\mathbf{W}$, *and vice versa.*

The proof of *Lemma 3.3* is given in Appendix A.2. This lemma allows us to obtain the eigenvalues of $\mathbf{S}$ by analyzing $\mathbf{W}$, which reduces the computational complexity.

3.5.2 Bounds for the Spectral Radius

Now, we give an upper bound for the spectral radius $\rho(\mathbf{T})$ in the next theorem, whose proof is given in Appendix A.3.

Theorem 3.1. *The spectral radius of the transition tensor* $\mathbf{T}$ *is upper-bounded by*

$$\rho(\mathbf{T}) \leq 1 - \min_{\hat{\alpha}}\{\mu_{\hat{\alpha}}\} + \max_{\hat{\alpha}}\{\rho(\theta_{\hat{\alpha}}^{\hat{\alpha}}\mathbf{A}_{\hat{\alpha}}^{\hat{\alpha}})\} + \rho(\hat{\boldsymbol{\theta}}), \tag{3.11}$$

where

$$\hat{\boldsymbol{\theta}} = \begin{bmatrix} 0 & \theta_{\hat{2}}^{\hat{1}} & \cdots & \theta_{\hat{M}}^{\hat{1}} \\ \theta_{\hat{1}}^{\hat{2}} & 0 & \cdots & \theta_{\hat{M}}^{\hat{2}} \\ \vdots & \vdots & \ddots & \vdots \\ \theta_{\hat{1}}^{\hat{M}} & \theta_{\hat{2}}^{\hat{M}} & \cdots & 0 \end{bmatrix}. \tag{3.12}$$

For a multilayer network, we define the dominating layer of infection as $\hat{\alpha}_I = arg\max_{\hat{\alpha}}\{\rho(\theta_{\hat{\alpha}}^{\hat{\alpha}}\mathbf{A}_{\hat{\alpha}}^{\hat{\alpha}})\}$ and the dominating layer of self-cure as $\hat{\alpha}_C = arg\min_{\hat{\alpha}}\{\mu_{\hat{\alpha}}\}$. Then, we can see that the upper bound in Eq. (3.11) depends on the dominating layers $\hat{\alpha}_I$ and $\hat{\alpha}_C$. This conclusion indicates that the dominating layers $\hat{\alpha}_I$ and $\hat{\alpha}_C$ play crucial roles in cascading failures. Other key parameters are the inter-layer infection rates. The complexity of direct calculation is polynomial w.r.t the matrix size, i.e., at $O((MN)^k)$; this bound simplifies the calculation to $O(\max\{MN^k, M^k\})$, where k is a positive integer.

Next, we give a lower bound in the following theorem, whose proof is given in Appendix A.4.

Theorem 3.2. *The spectral radius of the transition tensor* $\mathbf{T}$ *is lower-bounded by*

$$\rho(\mathbf{T}) \geq \max_{i,\hat{\alpha}}\{1 - \mu_{\hat{\alpha}}, \rho(\theta_{\hat{\alpha}}^{\hat{\alpha}}\mathbf{A}_{\hat{\alpha}}^{\hat{\alpha}}), \rho(\hat{\boldsymbol{\theta}}_{\boldsymbol{i}})\}, \tag{3.13}$$

where $\hat{\boldsymbol{\theta}}_{\boldsymbol{i}}$ *equals the Hadamard product of* $\hat{\boldsymbol{\theta}}$ *and the inter-layer adjacency tensor of Type-i object* $\mathbf{A}_{\mathbf{int}}(i)$*, i.e.,* $\hat{\boldsymbol{\theta}}_{\boldsymbol{i}} = \hat{\boldsymbol{\theta}} * \mathbf{A}_{\mathbf{int}}(i)$.

Similar to the dominating layer, we define the dominating type of objects as $i_d = arg\max_i\{\rho(\hat{\boldsymbol{\theta}}_{\boldsymbol{i}})\}$. We can see that the lower bound is determined by the spectral radius of dominating layers $\hat{\alpha}_I, \hat{\alpha}_C$ and the dominating type i_d. This lower bound reduces the complexity from $O((MN)^k)$ to $O(\max\{MN^k, NM^k\})$ compared to the direct calculation.

3.5.3 Approximations in Special Cases

In this subsection, we investigate the approximations of spectral radius in two special cases: 1) intra-layer propagation dominates ($\theta_{\hat{\alpha}}^{\hat{\alpha}} \gg \theta_{\hat{\beta}}^{\hat{\alpha}}$), which means that the cascading failure is easier to happen inside the same layer; and 2) inter-layer propagation dominates ($\theta_{\hat{\alpha}}^{\hat{\alpha}} \ll \theta_{\hat{\beta}}^{\hat{\alpha}}$), which means that the cascading failure is easier to happen across layers.

Theorem 3.3. *When $\theta_{\hat{\alpha}}^{\hat{\alpha}} \gg \theta_{\hat{\beta}}^{\hat{\alpha}}$ for $\hat{\beta} \neq \hat{\alpha}$, the spectral radius of the transition tensor $\mathbf{T}$ can be approximated by*

$$\rho(\mathbf{T}) \approx \max_{\hat{\alpha}}\{1 - \mu_{\hat{\alpha}} + \theta_{\hat{\alpha}}^{\hat{\alpha}} \rho(\mathbf{A}_{\hat{\alpha}}^{\hat{\alpha}})\}. \tag{3.14}$$

The proof is in Appendix A.5. This theorem implies that the dominating layer of the system will determine the system stability if the intra-layer propagation dominates.

Theorem 3.4. *When $\theta_{\hat{\alpha}}^{\hat{\alpha}} \ll \theta_{\hat{\beta}}^{\hat{\alpha}}$ for $\hat{\beta} \neq \hat{\alpha}$, the spectral radius of the transition tensor $\mathbf{T}$ can be approximated by*

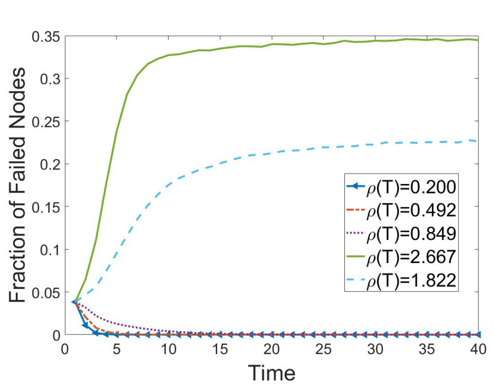
$$\rho(\mathbf{T}) \approx \max_{i}\{\rho(\hat{\boldsymbol{\theta}}_{\boldsymbol{i}} + \boldsymbol{\phi}\mathbf{I_M})\}, \tag{3.15}$$

where $\boldsymbol{\phi} = [1 - \mu_{\hat{1}}, \cdots, 1 - \mu_{\hat{M}}]$ and $\hat{\boldsymbol{\theta}}_{\boldsymbol{i}} = \hat{\boldsymbol{\theta}} * \mathbf{A_{int}}(i)$.

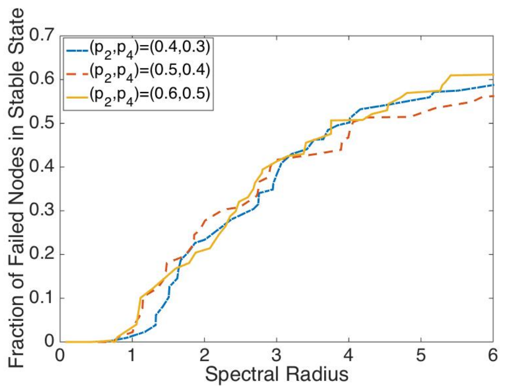
The proof is in Appendix A.6. This theorem implies that the dominating type of objects will determine the system stability if the inter-layer propagation dominates.

3.6 Numerical Analysis

In our simulation, the intra-layer connections are generated by following different *Erdös − Rènyi* random graphs [56], and the epidemic parameters are generated by following the uniform distribution. The initial failed nodes are randomly picked from the whole network.



(a) Fraction of failed nodes with different spectral radius over time



(b) Fraction of failed nodes at stable state

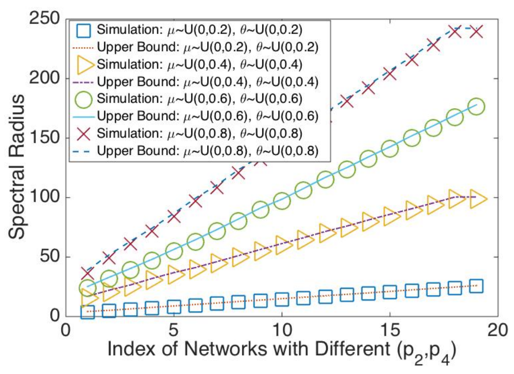
Figure 3.2: Influences of Spectral Radius

3.6.1 Influence of Spectral Radius

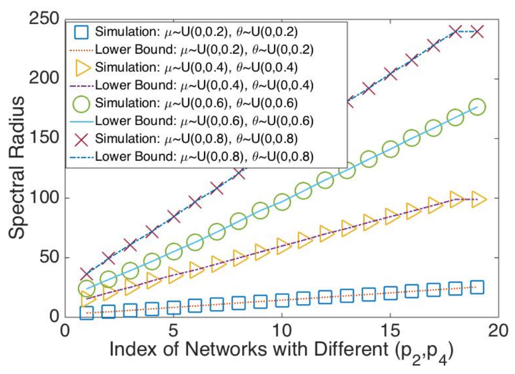
We first verify the failure threshold derived in Section 3.4 and explore more about how the spectral radius influences the system stability. We consider a network with 15 vehicles, 5 substations, 3 main stations, and 10 base stations. The inter-layer connections in layers 2, 4 follow the ER graph model $ER(n, p)$, where n is the number of nodes and p is the probability that two nodes are connected. From the definition of our multilayer network, $n = 33$ for all layers. Then, layers 2,4 follow $ER(33, p_2)$ and $ER(33, p_4)$, respectively.

We simulate how the fraction of failed nodes changes over time in systems with different spectral radii. We set $p_2 = 0.5$ and $p_4 = 0.4$. Initially there are 10 projected nodes failed. We repeat the simulation for 1000 times, and the averaged result is shown as Fig. 3.2(a). It is observed that the fraction of failed nodes will decrease to zero from the initial state when the spectral radius is smaller than 1. If the spectral radius is larger than 1, the fraction of failed nodes will change monotonically and then coverage to a positive value.

We also test how the fraction of failed nodes at stable state changes as the spectral radius increases. We test the results in three systems with different ER graph models in Fig. 3.2(b). We change the value of spectral radius by changing the infection rates and self-cure rates. We observe that the faction of failed nodes at stable state increases as spectral radius of transition tensor grows. That is, the system is less stable if the spectral radius of transition



(a) Performance of upper bound



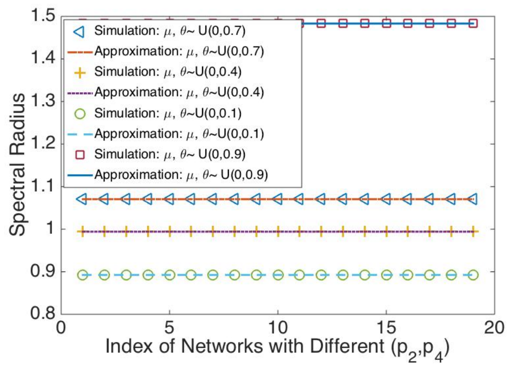
(b) Performance of lower bound

Figure 3.3: Performances of Bounds

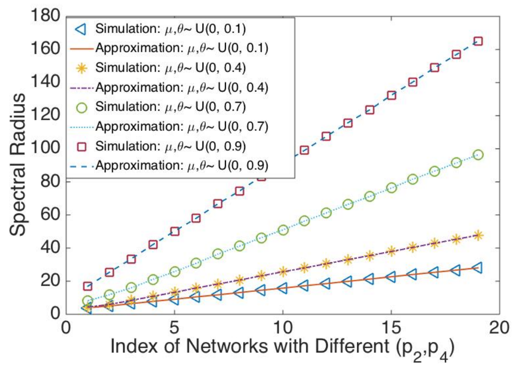
tensor gets larger. Note that the fraction of failed nodes is 0 when the spectral radius is less than 1, which agrees with Fig. 3.2(a). This conclusion shows that the spectral radius is a practical failure indicator to measure the system stability when designing robust systems.

3.6.2 Performances of Bounds and Approximations

To measure the performance of our bounds and approximations, we compare the approximated results with direct calculation results. Here, we consider 150 vehicles, 50 substations, 30 main stations, and 100 base stations. The results of the upper bound are shown in Fig. 3.3(a) and the results of the lower bound are given in Fig. 3.3(b). The special cases are shown in Fig. 3.4(a) and Fig. 3.4(b). The continuous curves are the approximated values and the discrete dots are the exact simulation results. We test the results in 20 sets of different multilayer networks with different ER graph setups of (p_2, p_4) from $(0, 0.05)$ to $(0.95, 1)$. The results are averaged for 10000 rounds. From the figures, we see that our bounds are tight. The approximation results in the special cases are also very close to the exact values.

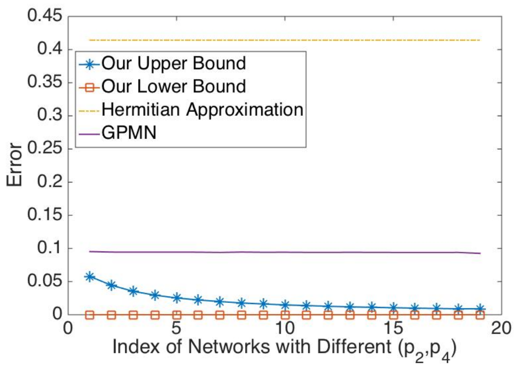


(a) Performances of approximations when inter-layer influence dominates

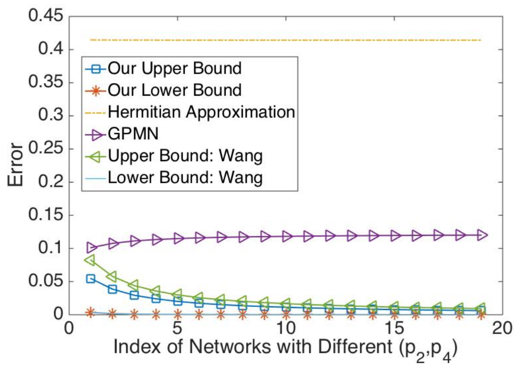


(b) Performances of approximations when intra-layer influence dominates

Figure 3.4: Performances of Approximations in Special Cases



(a) Seven-layer case



(b) Two-layer case

Figure 3.5: Comparison with Other Approximation Methods

3.6.3 Comparison with Other Approximations

To measure the performance of our approximation, we compare the results with other approximations from [46], [57] and [58] which are either tensors or matrices. Specifically, [46] gave the bounds for the two-layer cases based on matrix representation. In [57], the author gave the lower bound approximation of the spectral radius based on Hermitian and skew-Hermitian matrices. In [58], the result were derived using a tensor based method also known as Graph Product Multilayer Networks (GPMN). We first compare the error $\frac{|\rho_{approx}-\rho_{exact}|}{\rho_{exact}}$ for the Hermitian method, GPMN and our methods in our seven-layer network model with different expected node degrees shown in Fig. 3.5(a). To compare with other matrix-based methods, we set parameters of layers $\{1, 2, 3, 6, 7\}$ in our seven-layer cases as zero to construct a two layer network. The results are given in Fig. 3.5(b).

Observed from Fig. 3.5(a) and Fig. 3.5(b), the Hermitian method always has the largest error. In addition, the GPMN is better than the Hermitian method but worse than our methods, due to its limitation in representing general network structures. From Fig. 3.5(b), we see that the lower bound of Wang's method is close to our lower bound, while our upper bound is better than its upper bound. Moreover, Wang's method limits the number of layers up to two, while we can handle arbitrary number of layers. From the simulation results, we see that our method has better performance in the seven-layer case. In the two-layer case, our approximation is slightly better than the matrix method; but we can deal with a general system with more than two layers.

3.7 Conclusions

In this chapter, we proposed a scalable tensor-based framework for analyzing a multilayer complex systems. Specifically, we analyzed the epidemic spread based on the SIS model within our framework. We derived the failure transition equations, and showed that the spectral radius of the transition tensor is a failure indicator with an explicit failure threshold to measure the system stability. To analyze the failure propagation and reduce the computational complexity, we derived the upper and lower bounds for the failure indicator, as well as approximations in the special cases, where either inter-layer propagation dominates or intra-layer dominates. Validated by the numerical analysis, our bounds are tight and the approximations are close to the exact values.

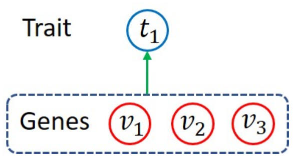
Chapter 4

Hypergraph Signal Processing: Theoretical Foundation and Practical Applications

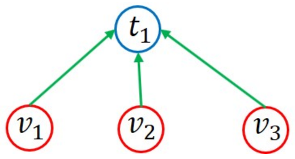
4.1 Introduction

Taking advantage of graph models in characterizing complex data structures, graph signal processing (GSP) has emerged as an exciting and promising new tool for processing large datasets with complex structures. A typical application of GSP is in image processing, where image pixels are modeled as graph signals embedding in nodes while pairwise similarities between pixels are captured by edges [60]. By modeling images using graphs, tasks such as image segmentation can take advantage of graph partition and GSP filters. Another example of GSP applications is in processing data from sensor networks [61]. Based on graph models directly built over network structures, a graph Fourier space could be defined according to the eigenspace of a representing graph matrix such as the Laplacian or adjacency matrix to

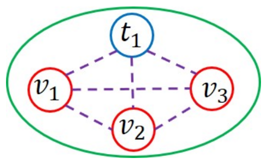
[2]Part of this chapter is reprinted, with permission, from [S. Zhang, Z. Ding, and S. Cui, "Introducing Hypergraph Signal Processing: Theoretical Foundation and Practical Applications", *IEEE Internet of Things Journal*, vol. 7, pp. 639 – 660, Jan. 2020.].



(a) Example of a trait and genes: the trait t_1 is triggered by three genes v_1, v_2, v_3 and the genes may also influence each other.



(b) Example of pairwise relationships: arrow links represent the influences from genes, whereas the potential interactions among genes cannot be captured.

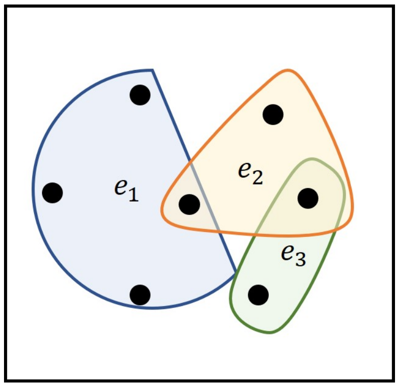


(c) Example of multi-lateral relationships: the solid circular line represents a quadrilateral relationship among four nodes, while the purple dash lines represent the potential inter-node interactions in this quadrilateral relationship.

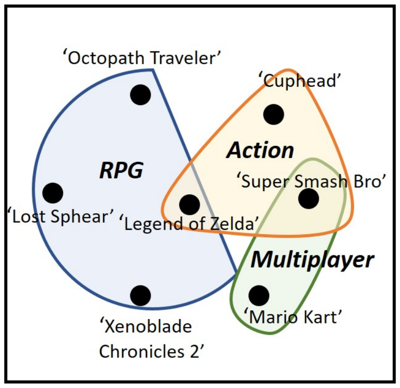
Figure 4.1: Example of Multi-lateral Relationships.

facilitate data processing operations such as denoising [15], filter banks [16] and compression [17].

Despite many demonstrated successes, the GSP defined over normal graphs also exhibits certain limitations. For example, normal graphs cannot capture high-dimensional interactions describing multi-lateral relationships among multiple nodes, which are critical for many practical applications. Since each edge in a normal graph only models the pairwise interactions between two nodes, the traditional GSP can only deal with the pairwise relationships defined by such edges. In reality, however, complex relationships may exist among a cluster of nodes, for which the use of pairwise links between every two nodes cannot capture their multi-lateral interactions [18]. In biology, for example, a trait may be attributed to multiple interactive genes [19] shown in Fig. 4.1(a), such that a quadrilateral interaction is more informative and powerful here. Another example is the social network with online social communities called folksonomies, where trilateral interactions occur among users, resources, and annotations [62, 63]. Thus, the traditional GSP based on matrix analysis has far been unable to efficiently handle such complex relationships. Clearly, there is a need for a more general graph model and graph signal processing concept to remedy the aforementioned shortcomings faced with the traditional GSP.



(a) Example of hypergraphs: the hyperedges are the overlaps covering nodes with different colors.



(b) A game dataset modeled by hypergraphs: each node is a specific game and each hyperedge is a catagory of games.

Figure 4.2: Hypergraphs and Applications.

To find a more general model for complex data structures, we venture into the area of high-dimensional graphs known as hypergraphs. The hypergraph theory is playing an increasingly important role in graph theory and data analysis, especially for analyzing high-dimensional data structures and interactions [64]. A hypergraph consists of nodes and hyperedges connecting more than two nodes [23]. As an example, Fig. 4.2(a) shows a hypergraph example with three hyperedges and seven nodes, whereas Fig. 4.2(b) provides a corresponding dataset modeled by this hypergraph. Indeed, a normal graph is a special case of a hypergraph, where each hyperedge degrades to a simple edge that only involves exactly two nodes.

Hypergraphs have found successes by generalizing normal graphs in many applications, such as clustering [24], classification [25], and prediction [26]. Moreover, a hypergraph is an alternative representation for a multi-layer network, and is useful when dealing with multi-tier relationships [65, 66]. Thus, a hypergraph is a natural extension of a normal graph in modeling signals of high-degree interactions. Presently, however, the literature provides little coverage on hypergraph signal processing (HGSP). The only known work [67] proposed a HGSP framework based on a special hypergraph called complexes. In this work [67], hypergraph signals are associated with each hyperedge, but its framework is limited to cell complexes, which cannot suitably model many real-world datasets and applications. Another shortcoming of the framework in [67] is the lack of detailed analysis and application examples to demonstrate its practicability. In addition, the attempt in [67] to extend some key concepts from the traditional GSP simply fails due to the difference in the basic setups between graph signals and hypergraph signals. In this chapter, we seek to establish a more general and practical HGSP framework, capable of handling arbitrary hypergraphs and naturally extending the traditional GSP concepts to handle the high-dimensional interactions. We will also provide real application examples to validate the effectiveness of the proposed framework.

Compared with the traditional GSP, a generalized HGSP faces several technical challenges. The first problem lies in the mathematical representation of hypergraphs. Developing an algebraic representation of a hypergraph is the foundation of HGSP. Currently there are two major approaches: matrix-based [68] and tensor-based [69]. The matrix-based method makes it hard to implement the hypergraph signal shifting while the tensor-based method is difficult to be understood conceptually. Another challenge is in defining signal shifting over the hyperedge. Signal shifting is easy to be defined as propagation along the link direction of a simple edge connecting two nodes in a regular graph. However, each hyperedge in hypergraphs involves more than two nodes. How to model signal interactions over a hyperedge requires careful considerations. Other challenges include the definition and interpretation of hypergraph frequency.

To address the aforementioned challenges and generalize the traditional GSP into a more general hypergraph tool to capture high dimension interactions, we propose a novel tensor-based HGSP framework in this chapter. The main contributions can be summarized as follows. Representing hypergraphs as tensors, we define a specific form of hypergraph signals and hypergraph signal shifting. We then provide an alternative definition of hypergraph Fourier space based on the orthogonal CANDECOMP/PARAFAC (CP) tensor decomposition, together with the corresponding hypergraph Fourier transform. To better interpret the hypergraph Fourier space, we analyze the resulting hypergraph frequency properties, including the concepts of frequency and bandlimited signals. Analogous to the traditional sampling theory, we derive the conditions and properties for perfect signal recovery from samples in HGSP. We also provide the theoretical foundation for the HGSP filter designs. Beyond these, we provide several application examples of the proposed HGSP framework:

1) We introduce a signal compression method based on the new sampling theory to show the effectiveness of HGSP in describing structured signals;

2) We apply HGSP in spectral clustering to show how the HGSP spectrum space acts as a suitable spectrum for hypergraphs;

3) We introduce a HGSP method for binary classification problems to demonstrate the practical application of HGSP in data analysis;

4) We introduce a filtering approach for the denoising problem to further showcase the power of HGSP;

5) Finally, we suggest several potential applicable background for HGSP, including Internet of Things (IoT), social network and nature language processing.

We compare the performance of HGSP-based methods with the traditional GSP-based methods and learning algorithms in all the above applications. All the features of HGSP make it an essential tool for IoT applications in the future.

We organize the rest of the chapter as follows. Section 4.2 first summarizes the preliminaries of the traditional GSP and hypergraphs. In Section 4.3, we then introduce the core definitions of HGSP, including the hypergraph signal, the signal shifting and the hypergraph Fourier space, followed by the frequency interpretation and decription of existing works in Section 4.4. We present some useful HGSP-based results such as the sampling theory and filter design in Section 4.5. With the proposed HGSP framework, we provide several potential applications of HGSP and demonstrate its effectiveness in Section 4.6, before presenting the final conclusions in Section 4.7.

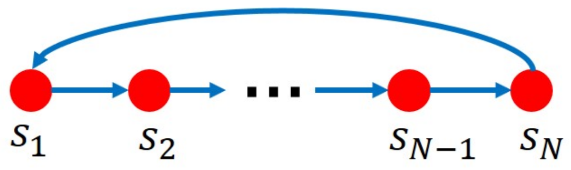


Figure 4.3: The Signal Shifting over Cyclic Graph.

4.2 Preliminaries

4.2.1 Overview of Graph Signal Processing

GSP is a recent tool used to analyze signals according to the graph models. Here, we briefly review the key relevant concepts of the traditional GSP [2, 13].

A dataset with N data points can be modeled as a normal graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ consisting of a set of N nodes $\mathcal{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_N\}$ and a set of edges $\mathcal{E}$. Each node of the graph $\mathcal{G}$ is a data point, whereas the edges describe the pairwise interactions between nodes. A graph signal represents the data associated with a node. For a graph with N nodes, there are N graph signals, which are defined as a signal vector $\mathbf{s} = [s_1 \quad s_2 \quad \dots \quad s_N]^{\mathrm{T}} \in \mathbb{R}^N$.

Usually, such a graph could be either described by an adjacency matrix $\mathbf{A_M} \in \mathbb{R}^{N\times N}$ where each entry indicates a pairwise link (or an edge), or by a Laplacian matrix $\mathbf{L_M} = \mathbf{D_M} - \mathbf{A_M}$ where $\mathbf{D_M} \in \mathbb{R}^{N\times N}$ is the diagonal matrix of degrees. Both the Laplacian matrix and the adjacency matrix can fully represent the graph structure. For convenience, we use a general matrix $\mathbf{F_M} \in \mathbb{R}^{N\times N}$ to represent either of them. Note that, since the adjacency matrix is eligible in both directed and undirected graph, it is more common in the GSP literatures. Thus, the generalized GSP is based on the adjacency matrix [13] and the representing matrix refers to the adjacency matrix in this chapter unless specified otherwise.

With the graph representation $\mathbf{F_M}$ and the signal vector $\mathbf{s}$, the graph shifting is defined as

$$\mathbf{s}' = \mathbf{F_M}\mathbf{s}. \tag{4.1}$$

Here, the matrix $\mathbf{F_M}$ could be interpreted as a graph filter whose functionality is to shift the signals along link directions. Taking the cyclic graph shown in Fig. 4.3 as an example, its adjacency matrix is a shifting matrix

$$\mathbf{F_M} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{4.2}$$

Typically, the shifted signal over the cyclic graph is calculated as $\mathbf{s}' = \mathbf{F_M}\mathbf{s} = [s_N \quad s_1 \quad \cdots \quad s_{N-1}]^{\mathrm{T}}$, which shifts the signal at each node to its next node.

The graph spectrum space, also called the graph Fourier space, is defined based on the eigenspace of $\mathbf{F_M}$. Assume that the eigen-decomposition of $\mathbf{F_M}$ is

$$\mathbf{F_M} = \mathbf{V_M^{-1}}\mathbf{\Lambda}\mathbf{V_M}. \tag{4.3}$$

The frequency components are defined by the eigenvectors of $\mathbf{F_M}$ and the frequencies are defined with respect to eigenvalues. The corresponding graph Fourier transform is defined as

$$\hat{\mathbf{s}} = \mathbf{V_M}\mathbf{s}. \tag{4.4}$$

With the definition of the graph Fourier space, the traditional signal processing and learning tasks, such as denoising [70] and classification [71], could be solved within the GSP framework. More details about the specific topics of GSP, such as the frequency analysis, filter design, and spectrum representation have been discussed in [61, 72, 73].

4.2.2 Introduction of Hypergraph

We begin with the definition of hypergraph and its possible representations.

Definition 1 (Hypergraph). *A general hypergraph $\mathcal{H}$ is a pair $\mathcal{H} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = \{\mathbf{v}_1, ..., \mathbf{v}_N\}$ is a set of elements called vertices and $\mathcal{E} = \{\mathbf{e}_1, ..., \mathbf{e}_K\}$ is a set of non-empty multi-element subsets of $\mathcal{V}$ called hyperedges. Let $M = \max\{|\mathbf{e}_i| : \mathbf{e}_i \in \mathcal{E}\}$ be the maximum cardinality of hyperedges, shorted as $m.c.e(\mathcal{H})$ of $\mathcal{H}$.*

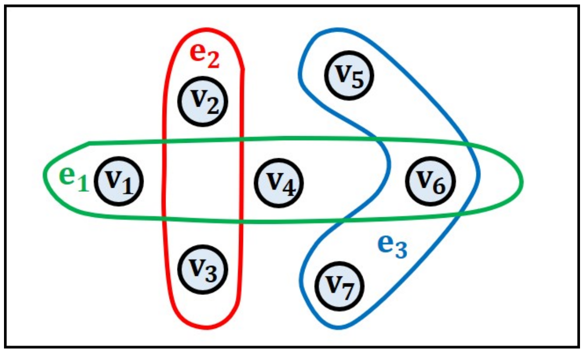


Figure 4.4: A hypergraph $\mathcal{H}$ with 7 nodes, 3 hyperedges and $m.c.e(\mathcal{H}) = 3$, where $\mathcal{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_7\}$ and $\mathcal{E} = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. Three hyperedges are $\mathbf{e}_1 = \{\mathbf{v}_1, \mathbf{v}_4, \mathbf{v}_6\}$, $\mathbf{e}_2 = \{\mathbf{v}_2, \mathbf{v}_3\}$ and $\mathbf{e}_3 = \{\mathbf{v}_5, \mathbf{v}_6, \mathbf{v}_7\}$.

In a general hypergraph $\mathcal{H}$, different hyperedges may contain different numbers of nodes. The $m.c.e(\mathcal{H})$ denotes the number of vertices in the largest hyperedge. An example of a hypergraph with 7 nodes, 3 hyperedges and $m.c.e = 3$ is shown in Fig. 4.4.

From the definition, we see that a normal graph is a special case of a hypergraph if $M = 2$. The hypergraph is a natural extension of the normal graph to represent high-dimensional interactions. To represent a hypergraph mathematically, there are two major methods based on matrix and tensor respectively. In the matrix-based method, a hypergraph is represented by a matrix $\mathbf{G} \in \mathbb{R}^{N \times E}$ where E equals the number of hyperedges. The rows of the matrix represent the nodes, and the columns represent the hyperedges [23]. Thus, each element in the matrix indicates whether the corresponding node is involved in the particular hyperedge. Although such a matrix-based representation is simple in formation, it is hard to define and implement signal processing directly as in GSP by using the matrix $\mathbf{G}$. Unlike the matrix-based method, tensor has better flexibility in describing the structures of the high-dimensional graphs [74]. More specifically, tensor can be viewed as an extension of matrix into high-dimensional domains. The adjacency tensor, which indicates whether nodes are connected, is a natural hypergraph counterpart to the adjacency matrix in the normal graph theory [75]. Thus, we prefer to represent the hypergraphs using tensors. In Section 4.3.1, we will provide more details on how to represent the hypergraphs and signals in tensor forms.

4.3 Definitions for Hypergraph Signal Processing

In this section, we introduce the core definitions used in our HGSP framework.

4.3.1 Algebraic Representation of Hypergraphs

The traditional GSP mainly relies on the representing matrix of a graph. Thus, an effective algebraic representation is also helpful in developing a novel HGSP framework. As we mentioned in Section 4.2, tensor is an intuitive representation for high-dimensional graphs. In this section, we introduce the algebraic representation of hypergraphs based on tensors.

Similar to the adjacency matrix whose 2-D entries indicate whether and how two nodes are pairwise connected by a simple edge, we adopt an adjacency tensor whose entries indicate whether and how corresponding subsets of M nodes are connected by hyperedges to describe hypergraphs [69].

Definition 2 (Adjacency tensor)**.** *A hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with N nodes and m.c.e$(\mathcal{H}) = M$ can be represented by an Mth-order N-dimension adjacency tensor $\mathbf{A} \in \mathbb{R}^{\overbrace{N\times N\times\cdots\times N}^{M\ times}}$ defined as*

$$\mathbf{A} = (a_{i_1 i_2 \cdots i_M}), \quad 1 \leq i_1, i_2, \cdots, i_M \leq N. \tag{4.5}$$

Suppose that $\mathbf{e}_l = \{\mathbf{v}_{l1}, \mathbf{v}_{l2}, \cdots, \mathbf{v}_{lc}\} \in \mathcal{E}$ is a hyperedge in $\mathcal{H}$ with the number of vertices $c \leq M$. Then, $\mathbf{e}_l$ is represented by all the elements $a_{p_1 \cdots p_M}$'s in $\mathbf{A}$, where a subset of c indices from $\{p_1, p_2, \cdots, p_M\}$ are exactly the same as $\{l_1, l_2, \cdots, l_c\}$ and the other $M-c$ indices are picked from $\{l_1, l_2, \cdots, l_c\}$ randomly. More specifically, these elements $a_{p_1 \cdots p_M}$'s describing

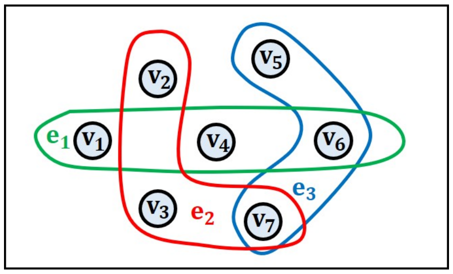
$\mathbf{e}_l$ *are calculated as*

$$a_{p_1 \cdots p_M} = c \left(\sum_{k_1, k_2, \cdots, k_c \geq 1, \sum_{i=1}^{c} k_i = M} \frac{M!}{k_1! k_2! \ldots k_c!} \right)^{-1}. \tag{4.6}$$

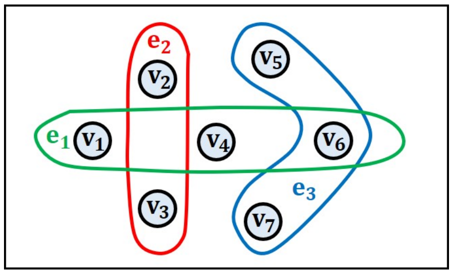
Meanwhile, the entries, which do not correspond to any hyperedge $\mathbf{e} \in \mathcal{E}$*, are zeros.*

Note that Eq. (4.6) enumerates all the possible combinations of c positive integers $\{k_1, \cdots, k_c\}$, whose summation satisfies $\sum_{i=1}^{c} k_i = M$. Obviously, when the hypergraph degrades to the normal graph with $c = M = 2$, the weights of edges are calculated as one, i.e., $a_{ij} = a_{ji} = 1$ for an edge $\mathbf{e} = (i, j) \in \mathcal{E}$. Then, the adjacency tensor is the same as the adjacency matrix. To understand the physical meaning of the adjacency tensor and its weight, we start with the M-uniform hypergraph with N nodes, where each hyperedge has exactly M nodes [76]. Since each hyperedge has an equal number of nodes, all hyperedges follow a consistent form to describe an M-lateral relationship with $m.c.e = M$. Obviously, such M-lateral relationships can be represented by an Mth-order tensor $\mathbf{A}$, where the entry $a_{i_1 i_2 \cdots i_M}$ indicates whether the nodes $\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \cdots, \mathbf{v}_{i_M}$ are in the same hyperedge, i.e., whether a hyperedge $\mathbf{e} = \{\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \cdots, \mathbf{v}_{i_M}\}$ exists. If the weight is nonzero, the hyperedge exists; otherwise, the hyperedge does not exist. Taking the 3-uniform hypergraph in Fig. 4.5(a) as an example, the hyperedge $\mathbf{e}_1$ is characterized by $a_{146} = a_{164} = a_{461} = a_{416} = a_{614} = a_{641} \neq 0$, the hyperedge $\mathbf{e}_2$ is characterized by $a_{237} = a_{327} = a_{732} = a_{723} = a_{273} = a_{372} \neq 0$, and $\mathbf{e}_3$ is represented by $a_{567} = a_{576} = a_{657} = a_{675} = a_{756} = a_{765} \neq 0$. All other entries in $\mathbf{A}$ are zero. Note that, all the hyperedges in an M-uniform hypergraph has the same weight. Different hyperedges are distinguished by the indices of the entries. More specifically, similarly as a_{ij} in the adjancency matrix implies the connection direction from node $\mathbf{v}_j$ to node $\mathbf{v}_i$ in GSP, an entry $a_{i_1 i_2 \cdots i_M}$ characterizes one direction of the hyperedge $\mathbf{e} = \{\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \cdots, \mathbf{v}_{i_M}\}$ with node $\mathbf{v}_{i_M}$ as the source and node $\mathbf{v}_{i_1}$ as the destination.

However, for a general hypergraph, different hyperedges may contain different numbers of nodes. For example, in the hypergraph of Fig. 4.5(b), the hyperedge $\mathbf{e}_2$ only contains two



(a) A 3-uniform hypergraph $\mathcal{H}$ with 7 nodes, 3 hyperedges and $m.c.e(\mathcal{H}) = 3$, where $\mathcal{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_7\}$ and $\mathcal{E} = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. Three hyperedges are $\mathbf{e}_1 = \{\mathbf{v}_1, \mathbf{v}_4, \mathbf{v}_6\}$, $\mathbf{e}_2 = \{\mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_7\}$ and $\mathbf{e}_3 = \{\mathbf{v}_5, \mathbf{v}_6, \mathbf{v}_7\}$.



(b) A general hypergraph $\mathcal{H}$ with 7 nodes, 3 hyperedges and $m.c.e(\mathcal{H}) = 3$, where $\mathcal{V} = \{\mathbf{v}_1, \cdots, \mathbf{v}_7\}$ and $\mathcal{E} = \{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$. Three hyperedges are $\mathbf{e}_1 = \{\mathbf{v}_1, \mathbf{v}_4, \mathbf{v}_6\}$, $\mathbf{e}_2 = \{\mathbf{v}_2, \mathbf{v}_3\}$ and $\mathbf{e}_3 = \{\mathbf{v}_5, \mathbf{v}_6, \mathbf{v}_7\}$.

Figure 4.5: Examples of Hypergraphs

nodes. How to represent the hyperedges with the number of nodes below $m.c.e = M$ may become an issue. To represent such a hyperedge $\mathbf{e}_l = \{\mathbf{v}_{l_1}, \mathbf{v}_{l_2}, ..., \mathbf{v}_{l_c}\} \in \mathcal{E}$ with the number of vertices $c < M$ in an Mth-order tensor, we can use entries $a_{i_1 i_2 \cdots i_M}$, where a subset of c indices are the same as $\{l_1, \cdots, l_c\}$ (possibly a different order) and the other $M - c$ indices are picked from $\{l_1, \cdots, l_c\}$ randomly. This process can be interpreted as generlaizing the hyperedge with c nodes to a hyperedge with M nodes by duplicating $M-c$ nodes from the set $\{\mathbf{v}_{l_1}, \cdots, \mathbf{v}_{l_c}\}$ randomly with possible repetitions. For example, the hyperedge $\mathbf{e}_2 = \{\mathbf{v}_2, \mathbf{v}_3\}$ in Fig. 4.5(b) can be represented by the entries $a_{233} = a_{323} = a_{332} = a_{322} = a_{223} = a_{232}$ in the third-order tensor $\mathbf{A}$, which could be interpreted as generalizing the original hyperedge with $c = 2$ to hyperedges with $M = 3$ nodes as Fig. 4.6. We can use Eq. (4.6) as a generalization coefficient of each hyperedge with respect to permutation and combination [69]. More specifically, for the adjacency tensor of the hypergraph in Fig. 4.5(b), the entries are calculated as $a_{146} = a_{164} = a_{461} = a_{416} = a_{614} = a_{641} = a_{567} = a_{576} = a_{657} = a_{675} = a_{756} = a_{765} = \frac{1}{2}$, $a_{233} = a_{323} = a_{332} = a_{322} = a_{223} = a_{232} = \frac{1}{3}$, where the remaining entries are set to zeros. Note that, the weight is smaller if the original hyperedge has fewer nodes in Fig. 4.5(b). More generally, based on the definition of adjacency tensor and Eq. (4.6), we can easily obtain the following property regarding the hyperedge weight.

Property 1. *Given two hyperedges* $\mathbf{e}_i = \{\mathbf{v}_1, \cdots, \mathbf{v}_I\}$ *and* $\mathbf{e}_j = \{\mathbf{v}_1, \cdots, \mathbf{v}_J\}$, *the edgeweight*

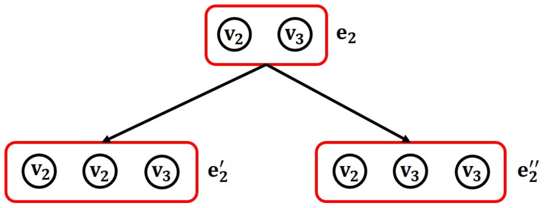


Figure 4.6: Interpretation of Generalizing $\mathbf{e}_2$ to Hyperedges with $M = 3$.

$w(\mathbf{e}_i)$ of $\mathbf{e}_i$ is different from the edgeweight $w(\mathbf{e}_j)$ of $\mathbf{e}_j$ in the adjacency tensor $\mathbf{A}$, i.e., $w(\mathbf{e}_i) \neq w(\mathbf{e}_j)$, if $I \neq J$. Moreover, $w(\mathbf{e}_i) = w(\mathbf{e}_j)$ iff $I = J$.

This property can help identify the length of each hyperedge based on the weights in the adjacency tensor. Moreover, the edgeweights of two hyperedges with the same number of nodes are the same. Different hyperedges with the same number of nodes are distinguished by their indices of entries in an adjacency tensor.

The degree $d(\mathbf{v}_i)$, of a vertex $\mathbf{v}_i \in \mathcal{V}$, is the number of hyperedges containing $\mathbf{v}_i$, i.e.,

$$d(\mathbf{v}_i) = \sum_{j_1,j_2\cdots,j_{M-1}=1}^{N} a_{ij_1j_2\cdots j_{M-1}}. \tag{4.7}$$

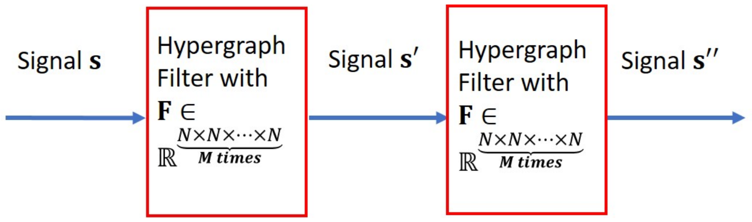
Then, the Laplacian tensor of the hypergraph $\mathcal{H}$ is defined as follows [69].

Definition 3 (Laplacian tensor)**.** *Given a hypergraph $\mathcal{H} = (\mathcal{V}, \mathcal{E})$ with N nodes and $m.c.e(\mathcal{H}) = M$, the Laplacian tensor is defined as*

$$\mathbf{L} = \mathbf{D} - \mathbf{A} \in \mathbb{R}^{\overbrace{N\times N\times\ldots\times N}^{M\ times}} \tag{4.8}$$

which is an Mth-order N-dimension tensor. Here, $\mathbf{D} = (d_{i_1 i_2\cdots i_M})$ is also an Mth-order N-dimension super-diagonal tensor with nonzero elements of $d_{\underbrace{ii\cdots i}_{M\ times}} = d(\mathbf{v}_i)$.

We see that both the adjacency and Laplacian tensors of a hypergraph $\mathcal{H}$ are super-symmetric. Moreover, when $m.c.e(\mathcal{H}) = 2$, they have similar forms to the adjacency and Laplacian matrices of undirected graphs respectively. Similar to GSP, we use an Mth-order N-dimension tensor $\mathbf{F}$ as a general representation of a given hypergraph $\mathcal{H}$ for convenience.



Signal $\mathbf{s}$
Hypergraph Filter with $\mathbf{F} \in \mathbb{R}^{\overbrace{N\times N\times\cdots\times N}^{M\ times}}$
Signal $\mathbf{s}'$
Hypergraph Filter with $\mathbf{F} \in \mathbb{R}^{\overbrace{N\times N\times\cdots\times N}^{M\ times}}$
Signal $\mathbf{s}''$

Figure 4.7: Signals in a Polynomial Filter.

As the adjacency tensor is more general, the representing tensor $\mathbf{F}$ refers to the adjacency tensor in this chapter unless specified otherwise.

4.3.2 Hypergraph Signal and Signal Shifting

Based on the tensor representation of hypergraphs, we now provide definitions for the hypergraph signal. In the traditional GSP, each signal element is related to one node in the graph. Thus, the graph signal in GSP is defined as an N-length vector if there are N nodes in the graph. Recall that the representing matrix of a normal graph can be treated as a graph filter, for which the basic form of the filtered signal is defined in Eq. (4.1). Thus, we could extend the definitions of the graph signal and signal shifting from the traditional GSP to HGSP based on the tensor-based filter implementation.

In HGSP, we also relate signal element to one node in the hypergraph. Naturally, we can define the original signal as an N-length vector if there are N nodes. Similarly as in GSP, we define the hypergraph shifting based on the representing tensor $\mathbf{F}$. However, since tensor $\mathbf{F}$ is of M-th order, we need an $(M-1)$-th order signal tensor to work with the hypergraph filter $\mathbf{F}$, such that the filtered signal is also an N-length vector as the original signal. For example, for a two-step polynomial filter shown as Fig. 4.7, the signals $\mathbf{s}, \mathbf{s}', \mathbf{s}''$ should all be in the same dimension and order. For the input and output signals in a HGSP system to have a consistent form, we define an alternative form of the hypergraph signal as below.

Definition 4 (Hypergraph signal). *For a hypergraph $\mathcal{H}$ with N nodes and $m.c.e(\mathcal{H}) = M$, an alternative form of hypergraph signal is an $(M-1)$-th order N-dimension tensor* $\mathbf{s}^{[M-1]}$

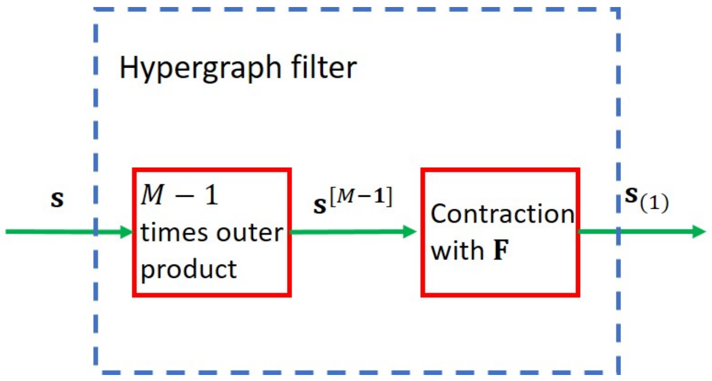


Figure 4.8: Diagram of Hypergraph Shifting.

obtained from $(M-1)$ *times outer product of the original signal* $\mathbf{s} = [s_1 \quad s_2 \quad \dots \quad s_N]^{\mathrm{T}}$, *i.e.,*

$$\mathbf{s}^{[M-1]} = \underbrace{\mathbf{s} \circ \dots \circ \mathbf{s}}_{M\text{-}1 \ times}, \tag{4.9}$$

where each entry in position $(i_1, i_2, \cdots, i_{M-1})$ *equals the product* $s_{i_1} s_{i_2} \cdots s_{i_{M-1}}$.

Note that the above hypergraph signal comes from the original signal. They are different forms of the same signal, which reflect the signal properties in different dimensions. For example, a second-order hypergraph signal highlights the properties of the two-dimensional signal components $s_i s_j$ while the original signal directly emphasizes more about the one-dimension properties. We will discuss in greater details on the relationship between the hypergraph signal and the original signal in Section 4.3.4.

With the definition of hypergraph signals, let us define the original domain of signals for convenience before we step into the signal shifting. Similarly as that the signals lie in the time domain for DSP, we have the following definition of hypergraph vertex domain.

Definition 5 (Hypergraph vertex domain)**.** *A signal lies in the hypergraph vertex domain if it resides on the structure of a hypergraph in the HGSP framework.*

The hypergraph vertex domain is a counterpart of time domain in HGSP. The signals are analyzed based on the structure among vertices in a hypergraph.

Next, we discuss how the signals shift on the given hypergraph. Recall that, in GSP, the signal shifting is defined by the product of the representing matrix $\mathbf{F_M} \in \mathbb{R}^{N\times N}$ and the signal vector $\mathbf{s} \in \mathbb{R}^N$, i.e., $\mathbf{s}' = \mathbf{F_M}\mathbf{s}$. Similarly, we define the hypergraph signal shifting based on its tensor $\mathbf{F}$ and the hypergraph signal $\mathbf{s}^{[M-1]}$.

Definition 6 (Hypergraph shifting)**.** *The basic shifting filter of hypergraph signals is defined as the direct contraction between the representing tensor* $\mathbf{F}$ *and the hypergraph signals* $\mathbf{s}^{[M-1]}$*, i.e.,*

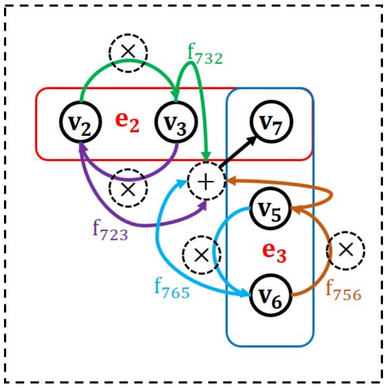
$$\mathbf{s}_{(1)} = \mathbf{F}\mathbf{s}^{[M-1]}, \tag{4.10}$$

where each element of the filter output is given by

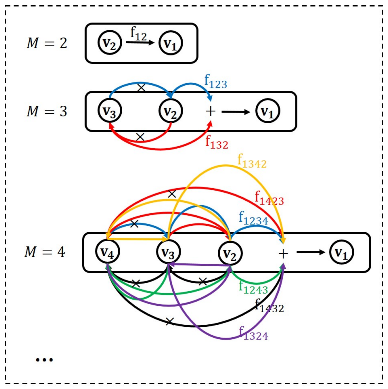
$$(\mathbf{s}_{(1)})_i = \sum_{j_1,\ldots,j_{M-1}=1}^{N} f_{ij_1\cdots j_{M-1}} s_{j_1} s_{j_2} \ldots s_{j_{M-1}}. \tag{4.11}$$

Since the hypergraph signal contracts with the representing tensor in $M-1$ order, the one-time filtered signal $\mathbf{s}_{(1)}$ is an N-length vector, which has the same dimension as the original signal. Thus, the block diagram of a hypergraph filter with $\mathbf{F}$ can be shown as Fig. 4.8.

Let us now consider the functionality of the hypergraph filter, as well as the physical insight of the hypergraph shifting. In GSP, the functionality of the filter $\mathbf{F_M}$ is simply to shift the signals along the link directions. However, interactions inside the hyperedge are more complex as it involves more than two nodes. In Eq. (4.11), we see that the filtered signal in $\mathbf{v}_i$ equals the summation of the shifted signal components in all hyperedges containing node $\mathbf{v}_i$, where $f_{ij_1\cdots j_{M-1}}$ is the weight for each involved hyperedge and $\{s_{j_1}, \cdots, s_{j_{M-1}}\}$ are the signals in the generalized hyperedges excluding s_i. Clearly, the hypergraph shifting multiplies signals in the same hyperedge of node $\mathbf{v}_i$ together before delivering the shift to a certain node $\mathbf{v}_i$. Taking the hypergraph in Fig. 4.5(a) as an example, node $\mathbf{v}_7$ is included in two hyperedges, $\mathbf{e}_2 = \{\mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_7\}$ and $\mathbf{e}_3 = \{\mathbf{v}_5, \mathbf{v}_6, \mathbf{v}_7\}$. According to Eq. (4.11), the



(a) Example of signal shifting to node $\mathbf{v}_7$. Different colors of arrows show the different directions of shifting; '$\times$' refers to multiplication and '$+$' refers to summation.



(b) Diagram of signal shifting to node $\mathbf{v}_1$ in an M-way hyperedge. Different colors refer to different shifting directions.

Figure 4.9: Diagram of Signal Shifting.

shifted signal in node $\mathbf{v}_7$ is calculated as

$$s_7 = f_{732} \times s_2 s_3 + f_{723} \times s_2 s_3 + f_{756} \times s_5 s_6 + f_{765} \times s_5 s_6, \tag{4.12}$$

where $f_{732} = f_{723}$ is the weight of the hyperedge $\mathbf{e}_2$ and $f_{756} = f_{765}$ is the weight for the hyperedge $\mathbf{e}_3$ in the adjacency tensor $\mathbf{F}$.

As the entry a_{ji} in the adjacency matrix of a normal graph indicates the link direction from the node $\mathbf{v}_i$ to the node $\mathbf{v}_j$, the entry $f_{i_1 \cdots i_M}$ in the adjacency tensor similarly indicates the order of nodes in a hyperedge as $\{\mathbf{v}_{i_M}, \mathbf{v}_{i_{M-1}}, \cdots \mathbf{v}_{i_1}\}$, where $\mathbf{v}_{i_1}$ is the destination and $\mathbf{v}_{i_M}$ is the source. Thus, the shifting by Eq. (4.12) could be interpreted as shown in Fig. 4.9(a). Since there are two possible directions from nodes $\{\mathbf{v}_2, \mathbf{v}_3\}$ to node $\mathbf{v}_7$ in $\mathbf{e}_2$, there are two components shifted to $\mathbf{v}_7$, i.e., the first two terms in Eq. (4.12). Similarly, there are also two components shifted by the hyperedge $\mathbf{e}_3$, i.e., the last two terms in Eq. (4.12). To illustrate the hypergraph shifting more explicitly, Fig. 4.9(b) shows a diagram of signal shifting to a certain node in an M-way hyperedge. From Fig. 4.9(b), we see that the graph shifting in GSP is a special case of the hypergraph shifting, where $M = 2$. Moreover, there are $K = (M-1)!$ possible directions for the shifting to one specific node in an M-way hyperedge.

4.3.3 Hypergraph Spectrum Space

We now provide the definitions of the hypergraph Fourier space, i.e., the hypergraph spectrum space. In GSP, the graph Fourier space is defined as the eigenspace of its representing matrix [61]. Similarly, we define the Fourier space of HGSP based on the representing tensor $\mathbf{F}$ of a hypergraph, which characterizes the hypergraph structure and signal shifting. For an M-th order N-dimension tensor $\mathbf{F}$, we can apply the orthogonal-CP decomposition [36] to write

$$\mathbf{F} \approx \sum_{r=1}^{R} \lambda_r \cdot \mathbf{f}_r^{(1)} \circ \ldots \circ \mathbf{f}_r^{(M)}, \tag{4.13}$$

with basis $\mathbf{f}_r^{(i)} \in \mathbb{R}^N$ for $1 \leq i \leq M$ and $\lambda_r \geq 0$. Since $\mathbf{F}$ is super-symmetric [37], i.e., $\mathbf{f}_r = \mathbf{f}_r^{(1)} = \mathbf{f}_r^{(2)} = \cdots = \mathbf{f}_r^{(M)}$, we have

$$\mathbf{F} \approx \sum_{r=1}^{R} \lambda_r \cdot \underbrace{\mathbf{f}_r \circ \ldots \circ \mathbf{f}_r}_{\text{M times}}. \tag{4.14}$$

Generally, we have the rank $R \leq N$ in a hypergraph. We will discuss how to construct the remaining $\mathbf{f}_i$, $R < i \leq N$, for the case of $R < N$ later in Section 4.3.6.

Now, by plugging Eq. (4.14) into Eq. (4.10), the hypergraph shifting can be written with the N basis $\mathbf{f}_i$'s as

$$\mathbf{s}_{(1)} = \mathbf{F}\mathbf{s}^{[M-1]} \tag{4.15a}$$

$$= (\sum_{r=1}^{N} \lambda_r \cdot \underbrace{\mathbf{f}_r \circ ... \circ \mathbf{f}_r}_{\text{M times}})(\underbrace{\mathbf{s} \circ ... \circ \mathbf{s}}_{\text{M-1 times}}) \tag{4.15b}$$

$$= \sum_{r=1}^{N} \lambda_r \mathbf{f}_r \underbrace{< \mathbf{f}_r, \mathbf{s} > \cdots < \mathbf{f}_r, \mathbf{s} >}_{\text{M-1 times}} \tag{4.15c}$$

$$= \underbrace{\begin{bmatrix} \mathbf{f}_1 & \cdots & \mathbf{f}_N \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix}}_{\text{iHGFT and filter in Fourier space}} \underbrace{\begin{bmatrix} (\mathbf{f}_1^{\mathrm{T}}\mathbf{s})^{M-1} \\ \vdots \\ (\mathbf{f}_N^{\mathrm{T}}\mathbf{s})^{M-1} \end{bmatrix}}_{\text{HGFT of the hypergraph signal}}, \tag{4.15d}$$

where $< \mathbf{f}_r, \mathbf{s} >= (\mathbf{f}_i^{\mathrm{T}}\mathbf{s})$ is the inner product between $\mathbf{f}_r$ and $\mathbf{s}$, and $(\cdot)^{M-1}$ is $(M-1)$th power.

From Eq. (4.15d), we see that the shifted signal in HGSP is in a similar decomposed to Eqs. (4.3) and (4.4) for GSP. The first two parts in Eq. (4.15d) work like $\mathbf{V_M^{-1}\Lambda}$ of the GSP eignen-decomposition, which could be interpreted as inverse Fourier transform and filter in the Fourier space. The third part can be understood as the hypergraph Fourier transform of the original signal. Hence, similarly as in GSP, we can define the hypergraph Fourier space and Fourier transform based on the orthogonal-CP decomposition of $\mathbf{F}$.

Definition 7 (Hypergraph Fourier space and Fourier transform)**.** *The hypergraph Fourier space of a given hypergraph $\mathcal{H}$ is defined as the space consisting of all orthogonal-CP decomposition basis $\{\mathbf{f}_1, \mathbf{f}_2, ..., \mathbf{f}_N\}$. The frequencies are defined with respect to the eigenvalue coefficients λ_i, $1 \leq i \leq N$. The hypergraph Fourier transform (HGFT) of hypergraph signals* *is defined as*

$$\hat{\mathbf{s}} = \mathcal{F}_C(\mathbf{s}^{[M-1]}) \tag{4.16a}$$

$$= \begin{bmatrix} (\mathbf{f}_1^{\mathrm{T}}\mathbf{s})^{M-1} \\ \vdots \\ (\mathbf{f}_N^{\mathrm{T}}\mathbf{s})^{M-1} \end{bmatrix}. \tag{4.16b}$$

Compared to GSP, if $M = 2$, the HGFT has the same form as the traditional GFT. In addition, since $\mathbf{f}_r$ is the orthogonal basis, we have

$$\mathbf{F}\mathbf{f}_r^{[M-1]} = \sum \lambda_i \mathbf{f}_i (\mathbf{f}_i^{\mathrm{T}}\mathbf{f}_r)^{M-1} = \lambda_r \mathbf{f}_r. \tag{4.17}$$

According to [37], a vector $\mathbf{x}$ is an E-eigenvector of an Mth-order tensor $\mathbf{A}$ if $\mathbf{A}\mathbf{x}^{[M-1]} = \lambda\mathbf{x}$ exists for a constant λ. Then, we obtain the following property of the hypergraph spectrum.

Property 2. *The hypergraph spectrum pair* $(\lambda_r, \mathbf{f}_r)$ *is an E-eigenpair of the representing tensor* $\mathbf{F}$*.*

Recall that the spectrum space of GSP is the eigenspace of the representing matrix $\mathbf{F_M}$. *Property 2* shows that HGSP has a consistent definition in the spectrum space as that for GSP.

4.3.4 Relationship between Hypergraph Signal and Original Signal

With HGFT defined, let us discuss more about the relationship between the hypergraph signal and the original signal in the Fourier space to understand the HGFT better. From

Eq. (4.16b), the hypergraph signal in the Fourier space is written as

$$\hat{\mathbf{s}} = \begin{bmatrix} (\mathbf{f}_1^{\mathrm{T}}\mathbf{s})^{M-1} \\ \vdots \\ (\mathbf{f}_N^{\mathrm{T}}\mathbf{s})^{M-1} \end{bmatrix}, \tag{4.18}$$

which can be further decomposed as

$$\hat{\mathbf{s}} = \underbrace{(\begin{bmatrix} \mathbf{f}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{f}_N^{\mathrm{T}} \end{bmatrix}\mathbf{s}) * (\begin{bmatrix} \mathbf{f}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{f}_N^{\mathrm{T}} \end{bmatrix}\mathbf{s}) * \cdots * (\begin{bmatrix} \mathbf{f}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{f}_N^{\mathrm{T}} \end{bmatrix}\mathbf{s})}_{M-1 \quad times}, \tag{4.19}$$

where $*$ denotes Hadamard product.

From Eq. (4.19), we see that the hypergraph signal in the hypergraph Fourier space is $M-1$ times Hadamard product of a component consisting of the hypergraph Fourier basis and the original signal. More specifically, this component works as the original signal in the hypergraph Fourier space, which is defined as

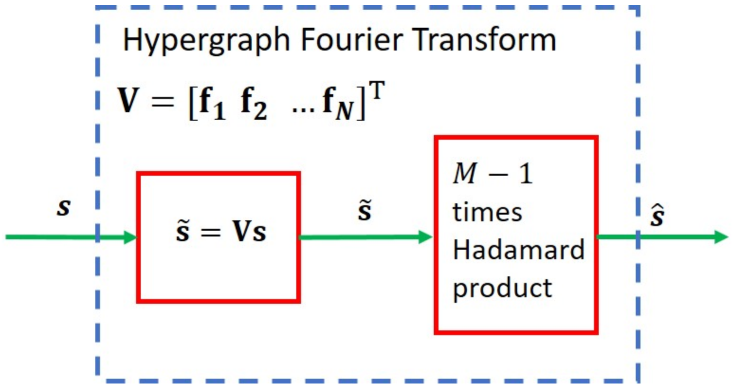
$$\tilde{\mathbf{s}} = \mathbf{V}\mathbf{s}, \tag{4.20}$$

where $\mathbf{V} = [\mathbf{f}_1 \quad \mathbf{f}_2 \quad \cdots \quad \mathbf{f}_N]^{\mathrm{T}}$ and $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$.

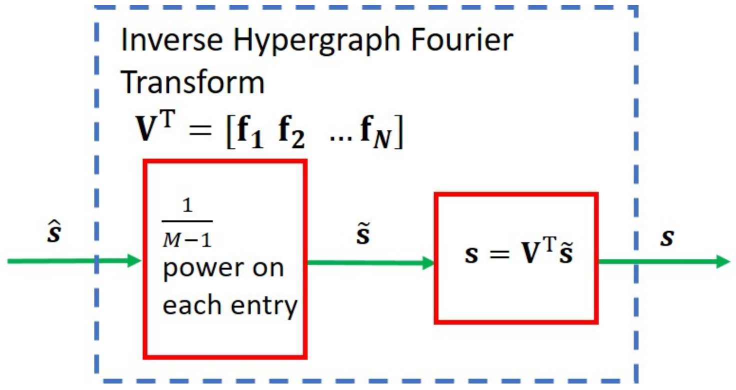
Recall the definitions of the hypergraph signal and vertex domain in Section 4.3.2, we have the following property.

Property 3. *The hypergraph signal is the $M-1$ times tensor outer product of the original signal in the hypergraph vertex domain, and the $M-1$ times Hadamard product of the original signal in the hypergraph frequency domain.*

Then, we could establish a connection between the original signal and the hypergraph signal in the hypergraph Fourier domain by the HGFT and inverse HGFT (iHGFT) as shown


Hypergraph Fourier Transform
$\mathbf{V} = [\mathbf{f}_1\ \mathbf{f}_2\ \dots \mathbf{f}_N]^{\mathrm{T}}$
$\mathbf{s}$
$\tilde{\mathbf{s}} = \mathbf{V}\mathbf{s}$
$\tilde{\mathbf{s}}$
$M-1$ times Hadamard product
$\hat{\mathbf{s}}$

(a) Process of HGFT


Inverse Hypergraph Fourier Transform
$\mathbf{V}^{\mathrm{T}} = [\mathbf{f}_1\ \mathbf{f}_2\ \dots \mathbf{f}_N]$
$\hat{\mathbf{s}}$
$\frac{1}{M-1}$ power on each entry
$\tilde{\mathbf{s}}$
$\mathbf{s} = \mathbf{V}^{\mathrm{T}}\tilde{\mathbf{s}}$
$\mathbf{s}$

(b) Process of iHGFT

Figure 4.10: Diagram of HGFT and iHGFT.

in Fig. 4.10. Such a relationship leads to some interesting properties and makes the HGFT implementation more straightforward, which will be further discussed in Section 4.3.6 and Section 4.3.7, respectively.

4.3.5 Hypergraph Frequency

As we now have a better understanding of the hypergraph Fourier space and Fourier transform, we can discuss more about the hypergraph frequency and its order. In GSP, the graph frequency is defined with respect to the eigenvalues of the representing matrix $\mathbf{F_M}$ and ordered by the total variation [61]. Similarly, in HGSP, we define the frequency relative to the coefficients λ_i from the orthogonal-CP decomposition. We order them by the total variation of frequency components $\mathbf{f}_i$ over the hypergraph. The total variation of a general signal component over a hypergraph is defined as follows.

Definition 8 (Total variation over hypergraph)**.** *Given a hypergraph $\mathcal{H}$ with N nodes and the normalized representing tensor $\mathbf{F}^{norm} = \frac{1}{\lambda_{max}}\mathbf{F}$, together with the original signal $\mathbf{s}$, the total variation over the hypergraph is defined as the total differences between the nodes and their corresponding neighbors in the perspective of shifting, i.e.,*

$$\mathbf{TV}(\mathbf{s}) = \sum_{i=1}^{N} |s_i - \sum_{j_1,\cdots,j_{M-1}=1}^{N} F^{norm}_{ij_1\cdots j_{M-1}} s_{j_1}...s_{j_{M-1}}| \tag{4.21a}$$

$$= ||\mathbf{s} - \mathbf{F}^{norm}\mathbf{s}^{[M-1]}||_1. \tag{4.21b}$$

We adopt the l_1-norm here only as an example of defining the total variation. Other norms may be more suitable depending on specific applications. Now, with the definition of total variation over hypergraphs, the frequency in HGSP is ordered by the total variation of the corresponding frequency component $\mathbf{f}_r$, i.e.,

$$\mathbf{TV}(\mathbf{f}_r) = ||\mathbf{f}_r - \mathbf{f}^{norm}_{r(1)}||_1, \tag{4.22}$$

where $\mathbf{f}^{norm}_{r(1)}$ is the output of one-time shifting for $\mathbf{f}_r$ over the normalized representing tensor.

From Eq. (4.21a), we see that the total variation describes how much the signal component changes from a node to its neighbors over the hypergraph shifting. Thus, we have the following definition of hypergraph frequency.

Definition 9 (Hypergraph frequency)**.** *Hypergraph frequency describes how oscillatory the signal component is with respect to the given hypergraph. A frequency component $\mathbf{f}_r$ is associated with a higher frequency if the total variation of this frequency component is larger.*

Note that, the physical meaning of graph frequency was stated in GSP [13]. Generally, the graph frequency is highly related to the total variation of the corresponding frequency component. Similarly, the hypergraph frequency also relates to the corresponding total variation. We will discuss more about the interpretation of the hypergraph frequency and its relationships with DSP and GSP later in Section 4.4.1, to further solidate our hypergraph frequency definition.

Based on the definition of total variation, we describe one important property of $\mathbf{TV}(\mathbf{f}_r)$ in the following theorem.

Theorem 4.1. *Define a supporting matrix*

$$\mathbf{P_s} = \frac{1}{\lambda_{\max}} \begin{bmatrix} \mathbf{f}_1 & \cdots & \mathbf{f}_N \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \begin{bmatrix} \mathbf{f}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{f}_N^{\mathrm{T}} \end{bmatrix}. \tag{4.23}$$

With the normalized representing tensor $\mathbf{F}^{norm} = \frac{1}{\lambda_{\max}}\mathbf{F}$*, the total variation of hypergraph spectrum* $\mathbf{f}_r$ *is calculated as*

$$\mathbf{TV}(\mathbf{f_r}) = ||\mathbf{f}_r - \mathbf{f}_{r(1)}^{norm}||_1, \tag{4.24a}$$

$$= ||\mathbf{f}_r - \mathbf{P_s}\mathbf{f}_r||_1, \tag{4.24b}$$

$$= |1 - \frac{\lambda_r}{\lambda_{\max}}|. \tag{4.24c}$$

Moreover, $\mathbf{TV}(\mathbf{f}_i) > \mathbf{TV}(\mathbf{f}_j)$ *iff* $\lambda_i < \lambda_j$*.*

Theorem 4.1 shows that the supporting matrix $\mathbf{P_s}$ can help us apply the total variation more efficiently in some real applications. Moreover, it provides the order of frequency according to the coefficients λ_i's with the following property.

Property 4. *A smaller* λ *is related to a higher frequency in the hypergraph Fourier space, where its corresponding spectrum basis is called a high frequency component.*

4.3.6 Signals with Limited Spectrum Support

With the order of frequency, we define the bandlimited signals as follows.

Definition 10 (Bandlimited signal)**.** *Order the coefficients as* $\lambda = [\lambda_1 \quad \cdots \quad \lambda_N]$ *where* $\lambda_1 \geq \cdots \geq \lambda_N \geq 0$*, together with their corresponding* $\mathbf{f}_r$*'s. A hypergraph signal* $\mathbf{s}^{[M-1]}$ *is defined as* K*-bandlimited if the HGFT transformed signal* $\hat{\mathbf{s}} = [\hat{s}_1, \cdots, \hat{s}_N]^{\mathrm{T}}$ *has* $\hat{s}_i = 0$ *for all* $i \geq K$ *where* $K \in \{1, 2, \cdots, N\}$*. The smallest* K *is defined as the bandwidth and the corresponding boundary is defined as* $W = \lambda_K$*.*

Note that, a larger λ_i corresponds to a lower frequency as we mentioned in *Property 4*. Then, the frequency are ordered from low to high in the definition above. Moreover, we use the index K instead of the coefficient value λ to define the bandwidth for the following reasons:

- Identical λ's in two diferent hypergraphs do not refer to the same frequency. Since each hypergraph has its own adjacency tensor and spectrum space, the comparison of multiple spectrum pairs $(\lambda_i, \mathbf{f}_i)$'s is only meaningful within the same hypergraph. Moreover, there exists a normalization issue in the decomposition of different adjacency tensors. Thus, it is not meaningful to compare λ_k's across two different hypergraphs.
- Since λ_k values are not continuous over k, different frequency cutoffs of λ may lead to the same bandlimited space. For example, suppose that $\lambda_k = 0.5$ and $\lambda_{k+1} = 0.8$. Then, $\lambda = 0.6$ and $\lambda' = 0.7$ would lead to the same cutoff in the frequency space, which makes bandwidth definition non-unique.

As we discussed in Section 4.3.4, the hypergraph signal is the Hadamard product of the original signal in the frequency domain. Then, we have the following property of bandwidth.

Property 5. *The bandwidth* K *is the same based on the HGFT of the hypergraph signals* $\hat{\mathbf{s}}$ *and that of the original signals* $\tilde{\mathbf{s}}$*.*

This property allows us to analyze the spectrum support of the hypergraph signal by looking into the original signal with lower complexity. Recall that we can add $\mathbf{f}_i$ by using zero coefficients λ_i when $R < N$ as mentioned in Section 4.3.3. The added basis should not affect the HGFT signals in Fourier space. According to the structure of bandlimited signal, we need the added $\mathbf{f}_i$ could meet the following conditions: (1) $\mathbf{f}_i \perp \mathbf{f}_p$ for $p \neq i$; (2) $\mathbf{f}_i^{\mathrm{T}} \cdot \mathbf{s} \rightarrow 0$; and (3) $|\mathbf{f}_i| = 1$.

4.3.7 Implementation and Complexity

We now consider the implementation and complexity issues of HGFT. Similar to GFT, the process of HGFT consists of two steps: decomposition and execution. The decomposition is to calculate the hypergraph spectrum basis, and the execution transforms signals from the hypergraph vertex domain into the spectrum domain.

- The calculation of spectrum basis by the orthogonal-CP decomposition is an important preparation step for HGFT. A straightforward algorithm would decompose the representing tensor $\mathbf{F}$ with the spectrum basis $\mathbf{f}_i$'s and coefficients λ_i's as in Eq. (4.14). Efficient tensor decomposition is an active topic in both fields of mathematics and engineering. There are a number of methods for CP decomposition in the literature. In [77], motivated by the spectral theorem for real symmetric matrices, orthogonal-CP decomposition algorithms for symmetric tensors are developed based polynomial equations. In [36], Afshar *et al.* proposed a more general decomposition algorithm for spatio-temporal data. Other works, including [78–80], tried to develop faster decomposition methods for signal processing and big data applications. The rapid development of tensor decomposition and the advancement of computation ability will benefit the efficient derivation of hypergraph spectrum.

- The execution of HGFT with a known spectrum basis is defined in Eq. (4.16b). According to Eq. (4.19), the HGFT of hypergraph signal is an $M-1$ times Hadamard product of the original signal in the hypergraph spectrum space. This relationship can help execute HGFT and iHGFT of hypergraph signals more efficiently by applying matrix operations on the original signals. Clearly, the complexity of calculating the original signals in the frequency domain $\tilde{\mathbf{s}} = \mathbf{V}\mathbf{s}$ is $O(N^2)$. In addition, since the computation complexity of the power function $x^{(M-1)}$ could be $O(\log(M-1))$ and each vector has N entries, the complexity of calculating the $M-1$ times Hadamard product is $O(N\log(M-1))$. Thus, the complexity of general HGFT implementation is $O(N^2 + N\log(M-1))$.

4.4 Discussions and Interpretations

In this section, we focus on the insights and physical meaning of frequency to help interpret the hypergraph spectrum space. We also consider the relationships between HGSP and other existing works to better understand the HGSP framework.

4.4.1 Interpretation of Hypergraph Spectrum Space

We are interested in an intuitive interpretation of the hypergraph frequency and its relations with the DSP and GSP frequencies. We start with the frequency and the total variation in DSP. In DSP, the discrete Fourier transform (DFT) of a sequence s_n is given by $\hat{s}_k = \sum_{n=0}^{N-1} s_n e^{-j\frac{2\pi kn}{N}}$ and the frequency is defined as $\nu_n = \frac{n}{N}$, $n = 0, 1, \cdots, N-1$. From [81], we can easily summarize the following conclusions:

- $\nu_n :\ 1 < n < \frac{N}{2} - 1$ corresponds to a continuous time signal frequency $\frac{n}{N}f_s$;
- $\nu_n :\ \frac{N}{2} + 1 < n < N-1$ corresponds to a continuous time signal frequency $-(1-\frac{n}{N})f_s$;
- $\nu_{\frac{N}{2}}$ corresponds to $f_s/2$;
- $n = 0$ corresponds to frequency 0.

Here, f_s is the critical sampling frequency. In traditional DFT, we generate the Fourier transform $\hat{f}(\omega) = \int_{-\infty}^{\infty} f(x)e^{-2\pi jx\omega}dx$ at each discrete frequency $\frac{n}{N}f_s$, $n = -\frac{N}{2}+1, -\frac{N}{2}+2, \cdots, \frac{N}{2}-1, \frac{N}{2}$. The highest and lowest frequencies correspond to $n = N/2$ and $n = 0$, respectively. Note that n varies from $-\frac{N}{2}+1$ to $\frac{N}{2}$ here. Since $e^{-j2\pi k\frac{n}{N}} = e^{-j2\pi k\frac{n+N}{N}}$, we can let n vary from 0 to $N-1$ and cover the complete period. Now, n varies in exact correspondence to ν_n, and the aforementioned conclusions are drawn. The highest frequency occurs at $n=\frac{N}{2}$.

The total variation in DSP is defined as the differences among the signals over time [82], i.e.,

$$\mathbf{TV}(\mathbf{s}) = \sum_{n=0}^{N-1} |s_n - s_{(n-1 \mod N)}| \tag{4.25a}$$

$$= ||\mathbf{s} - \mathbf{C_N s}||_{\mathbf{1}}, \tag{4.25b}$$

where

$$\mathbf{C_N} = \begin{bmatrix} 0 & 0 & \cdots & 0 & 1 \\ 1 & 0 & \cdots & 0 & 0 \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & \ddots & 0 & 0 \\ 0 & 0 & \cdots & 1 & 0 \end{bmatrix}. \tag{4.26}$$

When we perform the eigen-decomposition of $\mathbf{C_N}$, we see that the eigenvalues are $\lambda_n = e^{-j\frac{2\pi n}{N}}$ with eigenvector $\mathbf{f}_n$, $0 \le n \le N-1$. More specifically, the total variation of the frequency component $\mathbf{f}_n$ is calculated as

$$\mathbf{TV}(\mathbf{f}_n) = |1 - e^{j\frac{2\pi n}{N}}|, \tag{4.27}$$

which increases with n for $n \le \frac{N}{2}$ before decreasing with n for $\frac{N}{2} < n \le N-1$.

Obviously, the total variations of frequency components have a one-to-one correspondence to frequencies in the order of their values. If the total variation of a frequency component is larger, the corresponding frequency with the same index n is higher. It also has clear physical meaning, i.e., a higher frequency component changes faster over time, which implies

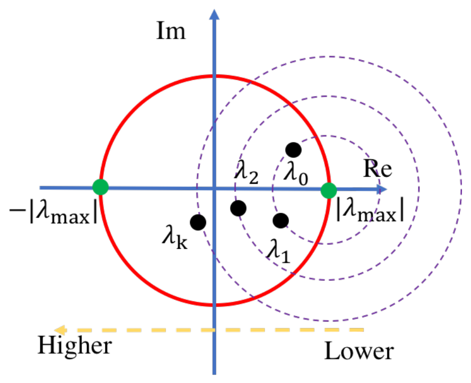


Figure 4.11: Example of Frequency Order in GSP for Complex Eigenvalues.

a larger total variation. Thus, we could also use the total variation of a frequency component to characterize its frequency in DSP.

Let us now consider the total variation and frequency in GSP, where the signals are analyzed in the graph vertex domain instead of the time domain. Similar to the fact that the frequency in DSP describes the rate of signal changes over time, the frequency in GSP illustrates the rate of signal changes over vertex [61]. Likewise, the total variation of the graph Fourier basis defined according to the adjacency matrix $\mathbf{F_M}$ could be used to characterize each frequency. Since GSP handles signals in the graph vertex domain, the total variation of GSP is defined as the differences between all the nodes and their neighbors, i.e.,

$$\mathbf{TV}(\mathbf{s}) = \sum_{n=1}^{N} |s_n - \sum_m {F_M}_{nm}^{norm} s_m| \tag{4.28a}$$

$$= ||\mathbf{s} - \mathbf{F_M}^{norm}\mathbf{s}||_1, \tag{4.28b}$$

where $\mathbf{F_M}^{norm} = \frac{1}{|\lambda_{max}|}\mathbf{F_M}$. If the total variation of the frequency component $\mathbf{f_M}_i$ is larger, it means the change over the graph between neighborhood vertices is faster, which indicates a higher graph frequency. Note that, once the graph is undirected, i.e., the eigenvalues are real numbers, the frequency decreases with the increase of the eigenvalue similar as HGSP in Section 4.3.5; otherwise, if the graph is undirected, i.e., the eigenvalues are complex, the frequency changes as shown in Fig. 4.11, which is consistency with the changing pattern of

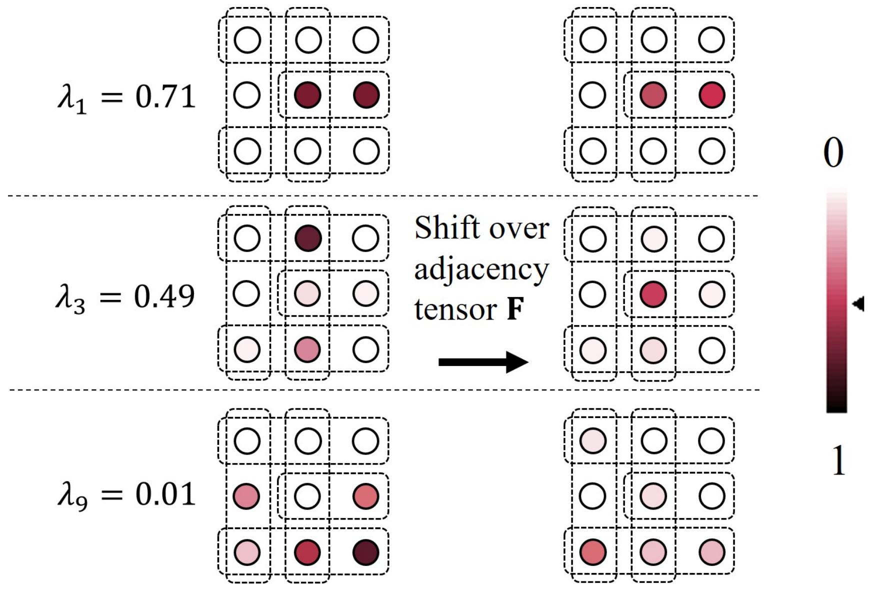


Figure 4.12: Frequency components in a hypergraph with 9 nodes and 5 hyperedges. The left panel shows the frequency components before shifting, and the right panel shows the frequency components after shifting. The values of signals are illustrated by colors. Higher frequency components imply larger changes across two panels.

DSP frequency [61].

We now turn to our HGSP framework. Like GSP, HGSP analyzes signals in the hypergraph vertex domain. Different from normal graphs, each hyperege in HGSP connects more than two nodes. The neighbors of a vertex $\mathbf{v}_i$ include all the nodes in the hyperedges containing $\mathbf{v}_i$. For example, if there exists a hyperedge $\mathbf{e}_1 = \{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$, nodes $\mathbf{v}_2$ and $\mathbf{v}_3$ are both neighbors of node $\mathbf{v}_1$. As we mentioned in Section 4.3.5, the total variation of HGSP is defined as the difference between continuous signals over the hypergraph, i.e., the difference between the signal components and their respective shifted versions:

$$\mathbf{TV}(\mathbf{s}) = \sum_{i=1}^{N} |s_i - \sum_{j_1,\ldots,j_{M-1}} F^{norm}_{ij_1\cdots j_{M-1}} s_{j_1} \cdots s_{j_{M-1}}| \tag{4.29a}$$

$$= ||\mathbf{s} - \mathbf{F}^{norm}\mathbf{s}^{[M-1]}||_1, \tag{4.29b}$$

where $\mathbf{F}^{norm} = \frac{1}{\lambda_{max}}\mathbf{F}$. Similar to DSP and GSP, pairs of $(\lambda_i, \mathbf{f}_i)$ in Eq. (4.14) characterize the hypergraph spectrum space. A spectrum component with a larger total variation represents a higher frequency component, which indicates faster changes over the hypergraph. Note that, as we mentioned in Section 4.3.5, the total variation is larger and the frequency is higher if the corresponding λ is smaller because we usually talk about undirected hypergraph and the λ's are real in the tensor decomposition. To illustrate it more clearly, we consider a hypergraph with 9 nodes, 5 hyperedges, and $m.c.e = 3$ as an example, shown in Fig. 4.12. As we mentioned before, a smaller λ indicates a higher frequency in HGSP. Hence, we see that the signals have more changes on each vertex if the frequency is higher.

4.4.2 Connections to other Existing Works

We now discuss the relationships between the HGSP and other existing works.

Graph Signal Processing

One of the motivations for developing HGSP is to develop a more general framework for signal processing in high-dimensional graphs. Thus, GSP should be a special case of HGSP. We illustrate the GSP-HGSP relationship as follows.

- Graphical models: GSP is based on normal graphs [13], where each simple edge connects exactly two nodes; HGSP is based on hypergraphs, where each hyperedge could connect more than two nodes. Clearly, the normal graph is a special case of hypergraph, for which the $m.c.e$ equals two. More specifically, a normal graph is a 2-uniform hypergraph [83]. Hypergraph provides a more general model for multi-lateral relationships while normal graphs are only able to model bilateral relationship. For example, a 3-uniform hypergraph is able to model the trilateral interaction among users in a social network [84]. As hypergraph is a more general model for high-dimensional interactions, HGSP is also more powerful for high-dimensional signals.

- Algebraic models: HGSP relies on tensors while GSP relies on matrices, which are second-order tensors. Benefiting from the generality of tensor, HGSP is broadly applicable in high-dimensional data analysis.

- Signals and signal shifting: In HGSP, we define the hypergraph signal as $M-1$ times tensor outer product of the original signal. More specifically, the hypergraph signal is the original signal if $M=2$. Basically, the hypergraph signal is the same as the graph signal if each hyperedge has exactly two nodes. Also shown in Fig. 4.9(b) of Section 4.3.3, graph shifting is a special case of hypergraph shifting when $M=2$.

- Spectrum properties: In HGSP, the spectrum space is defined over the orthogonal-CP decomposition in terms of the basis and coefficients, which are also the E-eigenpairs of the representing tensor [85], shown in Eq. (4.17). In GSP, the spectrum space is defined as the matrix eigenspace. Since the tensor algebra is an extension of matrix, the HGSP spectrum is also an extension of the GSP spectrum. For example, as discussed in Section 4.3, GFT is the same as HGFT when $M=2$.

Overall, HGSP is an extension of GSP, which is both more general and novel. The purpose of developing the HGSP framework is to facilitate more interesting signal processing tasks that involve high-dimensional signal interactions.

Higher-Order Statistics

Higher-order statistics (HOS) has been effectively applied in signal processing[86, 87], which can analyze the multi-lateral interactions of signal samples and have found successes in many applications, such as blind feature detection [88], decision [89], and signal classifications [90]. In HOS, the kth-order cumulant of random variables $\mathbf{x}=[x_1,\cdots,x_k]^{\mathrm{T}}$ is defined [91] based on the coefficients of $\mathbf{v}=[v_1,\cdots,v_k]^{\mathrm{T}}$ in the Talyor series expansion of cumulant-gernerating function, i.e.,

$$K(\mathbf{v})=\ln\mathbf{E}\{\exp(j\mathbf{v}^{\mathrm{T}}\mathbf{x})\}. \tag{4.30}$$

It is easy to see that HGSP and HOS are related in high-dimensional signal processing. They can be both represented by tensor. For example, in the multi-channel problems of [92], the 3rd-order cumulant $\mathbf{C} = \{C_{y_i,y_j,y_z}(t, t_1, t_2)\}$ of zero-mean signals can be represented as a multilinear array, e.g.,

$$C_{y_i,y_j,y_z}(t, t_1, t_2) = \mathbf{E}\{y_i(t)y_j(t + t_1)y_z(t + t_2)\}, \tag{4.31}$$

which is essentially a third-order tensor. More specifically, if there are k samples, the cumulant $\mathbf{C}$ can be represented as an p^k-element vector, which is the flattened signal tensor similar to the n-mode flattening of HGSP signals.

Although both HOS and HGSP are high-dimensional signal processing tools, they focus on complementary aspects of the signals. Specifically, HGSP aims to analyze signals over the high-dimensional vertex domain, while HOS focuses on the statistical domain. In addition, the forms of signal combination are also different, where HGSP signals are based on the hypergraph shifting defined as in Eq. (4.11), whereas HOS cumulants are based on the statistical average of shifted signal products.

Learning over Hypergraphs

Hypergraph learning is another tool to handle structured data and sometimes uses similar techniques to HGSP. For example, the authors of [93] proposed an alternative definition of hypergraph total variation and design algorithms in accordance for classification and clustering problems. In addition, hypergraph learning also has its own definition of the hypergraph spectrum space. For example, [24, 63] represented the hypergraphs using a graph-like similarity matrix and defined a spectrum space as the eigenspace of this similarity matrix. Other works considered different aspects of hypergraph, including the hypergraph Laplacian [94] and hypergraph lifting [20].

The HGSP framework exhibits features different from hypergraph learning: 1) HGSP defines a framework that generalizes the classical digital signal processing and traditional graph signal processing; 2) HGSP applies different definitions of hypergraph characteristics such as the total variation, spectrum space, and Laplacian; 3) HGSP cares more about the spectrum space while learning focuses more on data; 4) As HGSP is an extension of DSP and GSP, it is more suitable to handle detailed tasks such as compression, denoising, and detection. All these features make HGSP a different technical concept from hypergraph learning.

4.5 Tools for Hypergrph Signal Processing

In this section, we introduce several useful tools built within the framework of HGSP.

4.5.1 Sampling Theory

Sampling is an important tool in data analysis, which selects a subset of individual data points to estimate the characteristics of the whole population [42]. Sampling plays an important role in applications such as compression [95] and storage [96]. Similar to sampling signals in time, the HGSP sampling theory can be developed to sample signals over the vertex domain. We now introduce the basics of HGSP sampling theory for lossless signal dimension reduction.

To reduce the size of a hypergraph signal $\mathbf{s}^{[M-1]}$, there are two main approaches: 1) to reduce the dimension of each order; and 2) to reduce the number of orders. Since the reduction of order breaks the structure of hypergraph and cannot always guarantee perfect recovery, we adopt the dimension reduction of each order. To change the dimension of a certain order, we can use the n-Mode product. Since each order of the hypergraph signal is equivalent, the n-Mode product operators of each order are the same. Then, the sampling operation of the hypergraph signal is defined as follows:

Definition 11 (Sampling and Interpolation)**.** *Suppose that Q is the dimension of each sam-*

pled order. The sampling operation is defined as

$$\mathbf{s}_{\mathbf{Q}}^{[\mathbf{M-1}]} = \mathbf{s}^{[\mathbf{M-1}]} \times_1 \mathbf{U} \times_2 \mathbf{U} \cdots \times_{M-1} \mathbf{U}, \tag{4.32}$$

where the sampling operator is $\mathbf{U} \in \mathbb{R}^{Q\times N}$ to be defined later, and the sampled signal is $\mathbf{s}_{\mathbf{Q}}^{[\mathbf{M-1}]} \in \mathbb{R}^{\overbrace{Q\times Q\times \ldots \times Q}^{M-1 \text{ times}}}$.

The interpolation operation is defined by

$$\mathbf{s}^{[\mathbf{M-1}]} = \mathbf{s}_{\mathbf{Q}}^{[\mathbf{M-1}]} \times_1 \mathbf{T} \times_2 \mathbf{T} \cdots \times_{M-1} \mathbf{T}, \tag{4.33}$$

where the interpolation operator is $\mathbf{T} \in \mathbb{R}^{N\times Q}$ to be defined later.

As presented in Section 4.3, the hypergraph signal and original signal are different forms of the same data. They may have similar properties in structures. To derive the sampling theory for perfect signal recovery efficiently, we first consider the sampling operations of the original signal.

Definition 12 (Sampling original signal)**.** *Suppose an original K-bandlimited signal $\mathbf{s} \in \mathbb{R}^N$ is to be sampled into $\mathbf{s}_{\mathbf{Q}} \in \mathbb{R}^Q$, where $q = \{q_1, \cdots, q_Q\}$ denotes the sequence of sampled indices and $q_i \in \{1, 2, \cdots, N\}$. The sampling operator $\mathbf{U} \in \mathbb{R}^{Q\times N}$ is a linearing mappling from $\mathbb{R}^N$ to $\mathbb{R}^Q$, defined by*

$$U_{ij} = \begin{cases} 1, & j = q_i; \\ 0, & \textit{otherwise}, \end{cases} \tag{4.34}$$

and the interpolation operator $\mathbf{T} \in \mathbb{R}^{N\times Q}$ is a linear mapping from $\mathbb{R}^Q$ to $\mathbb{R}^N$. Then, the sampling operation is defined by

$$\mathbf{s}_{\mathbf{Q}} = \mathbf{U} \cdot \mathbf{s}, \tag{4.35}$$

and the interpolation operation is defined by

$$\mathbf{s}' = \mathbf{T} \cdot \mathbf{s}_{\mathbf{Q}}. \tag{4.36}$$

Analyzing the structure of the sampling operations, we have the following properties.

Theorem 4.2. *The hypergraph signal* $\mathbf{s}^{[\mathbf{M-1}]}$ *shares the same sampling operator* $\mathbf{U} \in \mathbb{R}^{Q\times N}$ *and interpolation operator* $\mathbf{T} \in \mathbb{R}^{N\times Q}$ *with the original signal* $\mathbf{s}$.

The proof is given in Appendix B.2. Given *Theorem 4.2*, we only need to analyze the operations of the original signal in the sampling theory. Next, we discuss the conditions for perfect recovery. For the original signal, we have the following property.

Lemma 4.1. *Suppose that* $\mathbf{s} \in \mathbb{R}^N$ *is a* K*-bandlimited signal. Then, we have*

$$\mathbf{s} = \mathcal{F}_{[K]}^{\mathrm{T}} \tilde{\mathbf{s}}_{[K]}, \tag{4.37}$$

where $\mathcal{F}_{[K]}^{\mathrm{T}} = [\mathbf{f}_1, \cdots, \mathbf{f}_K]$ *and* $\tilde{\mathbf{s}}_{[K]} \in \mathbb{R}^K$ *consists of the first* K *elements of the original signal in the frequency domain, i.e.,* $\tilde{\mathbf{s}}$.

This lemma implies that the first K frequency components carry all the information of the original signal. Since the hypergraph signal and the original signal share the same sampling operators, we can reach a similar conclusion for perfect recovery as [95, 97], given in the following theorem.

Theorem 4.3. *Define the sampling operator* $\mathbf{U} \in \mathbb{R}^{Q\times N}$ *according to* $U_{ji} = \delta[i - q_j]$ *where* $1 \le q_i \le N$, $i = 1, \ldots, Q$. *By choosing* $Q \ge K$ *and the interpolation operator* $\mathbf{T} = \mathcal{F}_{[K]}^{\mathrm{T}} \mathbf{Z} \in \mathbb{R}^{N\times Q}$ *with* $\mathbf{Z}\mathbf{U}\mathcal{F}_{[K]}^{\mathrm{T}} = \mathbf{I_K}$ *and* $\mathcal{F}_{[K]}^{\mathrm{T}} = [\mathbf{f}_1, \cdots, \mathbf{f}_K]$, *we can achieve a perfect recovery, i.e.,* $\mathbf{s} = \mathbf{TUs}$ *for all* K*-bandlimited original signal* $\mathbf{s}$ *and the corresponding hypergraph signal* $\mathbf{s}^{[M-1]}$.

The proof is provided in Appendix B.4. *Theorem 4.3* shows that a perfect recovery is possible for a bandlimited hypergraph signal. We now examine some interesting properties of the sampled signal.

From the previous discussion, we have $\tilde{\mathbf{s}}_{[K]} = \mathbf{Z}\mathbf{s}_{\mathbf{Q}}$, which has a similar form to HGFT, where $\mathbf{Z}$ can be treated as the Fourier transform operator. Suppose that $Q = K$ and $\mathbf{Z} = [\mathbf{z}_1 \quad \cdots \quad \mathbf{z}_K]^{\mathrm{T}}$. We have the following first-order difference property.

Theorem 4.4. *Define a new hypergraph by* $\mathbf{F_K} = \sum_{i=1}^{K} \lambda_i \cdot \mathbf{z}_i \circ \cdots \circ \mathbf{z}_i$. *Then, for all* K*-bandlimited signal* $\mathbf{s}^{[M-1]} \in \mathbb{R}^{\overbrace{\scriptstyle N\times N\times\ldots\times N}^{}_{M\ times}}$, *it holds that*

$$\mathbf{s}_{[K]} - \mathbf{F_K}\mathbf{s}_{[K]}^{[M-1]} = \mathbf{U}(\mathbf{s} - \mathbf{F}\mathbf{s}^{[M-1]}). \tag{4.38}$$

Theorem 4.4 shows that the sampled signals form a new hypergraph that preserves the information of the one-time shifting filter over the original hypergraph. For example, the left-hand side of Eq. (4.38) represent the difference between the sampled signal and the one-time shifted version in the new hypergraph. The right-hand side of Eq. (4.38) is the difference between a signal and its one-time shifted version in the original hypergraph, together with the sampling operator. That is, the sampled result of the one-time shifting differences in the original hypergraph is equal to the one-time shifting differences in the new sampled hypergraph.

4.5.2 Filter Desgin

Filter is an important tool in signal processing applications such as denoising, feature enhancement, smoothing, and classification. In GSP, the basic filtering is defined as $\mathbf{s}' = \mathbf{F_M}\mathbf{s}$ where $\mathbf{F_M}$ is the representing matrix [13]. In HGSP, the basic hypergraph filtering is defined in Section 4.3.3 as $\mathbf{s}_{(1)} = \mathbf{F}\mathbf{s}^{[M-1]}$, which is designed according to the tensor contraction. The HGSP filter is a multilinear mapping [98]. The high-dimensionality of tensors provides more flexibility in designing the HGSP filter.

Polynomial Filter based on Representing Tensor

Polynomial filter is one basic form of HGSP filters, with which signals are shifted several times over the hypergraph. An example of polynomial filter is given as Fig. 4.7 in Section

4.3.2. A k-time shifting filter is defined as

$$\mathbf{s}_{(k)} = \mathbf{F}\mathbf{s}_{(k-1)}^{[M-1]} \tag{4.39a}$$

$$= \underbrace{\mathbf{F}(\mathbf{F}(...(\mathbf{F}\mathbf{s}^{[M-1]})^{[M-1]})^{[M-1]})^{[M-1]}}_{k \quad times}. \tag{4.39b}$$

More generally, a polynomial filter is designed as

$$\mathbf{s}' = \sum_{k=1}^{a} \alpha_k \mathbf{s}_{(k)}, \tag{4.40}$$

where $\{\alpha_k\}$ are the filter coefficients. Such HGSP filters are based on multilinear tensor contraction, which could be used for different signal processing tasks by selecting specific parameters a and $\{\alpha_i\}$.

In addition to the general polynomial filter based on hypergraph signals, we provide another specific form of polynomial filter based on the original signals. As mentioned in Section 4.3.5, the supporting matrix $\mathbf{P_s}$ in Eq. (4.23) captures all the information of the frequency space. For example, the unnormalized supporting matrix $\mathbf{P} = \lambda_{max}\mathbf{P_s}$ is calculated as

$$\mathbf{P} = \begin{bmatrix} \mathbf{f}_1 & \cdots & \mathbf{f}_N \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \begin{bmatrix} \mathbf{f}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{f}_N^{\mathrm{T}} \end{bmatrix}. \tag{4.41}$$

Obviously, the hypergraph spectrum pair $(\lambda_r, \mathbf{f}_r)$ is an eigenpair of the supporting matrix $\mathbf{P}$. Moreover, *Theorem 4.1* shows that the total variation of frequency component equals to a function of $\mathbf{P}$, i.e.,

$$\mathbf{TV}(\mathbf{f}_r) = ||\mathbf{f}_r - \frac{1}{\lambda_{max}}\mathbf{P}\mathbf{f}_r||_1. \tag{4.42}$$

From Eq. (4.42), $\mathbf{P}$ can be interpreted as a shifting matrix for the original signal. Accordingly, we can design a polynomial filter for the original signal based on the supporting matrix

P whose kth-order term is defined as

$$\mathbf{s}_{<k>} = \mathbf{P}^k\mathbf{s}. \tag{4.43}$$

The a-th order polynomial filter is simply given as

$$\mathbf{s}' = \sum_{k=1}^{a} \alpha_k \mathbf{P}^k \mathbf{s}. \tag{4.44}$$

A polynomial filter over the original signal can be determined with specific choices of a and α.

Let us consider some interesting properties of the polynomial filter for the original signal. First, given the kth-order term, we have the following property as *Lemma 4.2*.

Lemma 4.2.

$$\mathbf{s}_{<k>} = \sum_{r=1}^{N} \lambda_r^k (\mathbf{f}_r^{\mathrm{T}}\mathbf{s})\mathbf{f}_r. \tag{4.45}$$

From *Lemma 4.2*, we obtain the following property of the polynomial filter for the original signal.

Theorem 4.5. *Let $h(\cdot)$ be a polynomial function. For the polynomial filter $\mathbf{H} = h(\mathbf{P})$ for the original signal, the filtered signal satisfies*

$$\mathbf{H}\mathbf{s} = \sum_{r=1}^{N} h(\mathbf{s}_{<i>}) = \sum_{r=1}^{N} h(\lambda_r)\mathbf{f}_r(\mathbf{f}_r^{\mathrm{T}}\mathbf{s}). \tag{4.46}$$

This theorem works as the invariance property of exponential in HGSP, similar to those in GSP and DSP [23]. Eq. (4.40) and Eq. (4.43) provide more choices for HGSP polynomial filters in hypergraph signal processing and data analysis. We will give specific examples of practical applications in Section 4.6.

General Filter Design based on Optimization

In GSP, some filters are designed via optimization formulations [13,71,99]. Similarly, general HGSP filters can also be designed via optimization approaches. Assume $\mathbf{y}$ is the oberserved signal before shifting and $\mathbf{s} = h(\mathbf{F}, \mathbf{y})$ is the shifted signal by HGSP filter $h(\cdot)$ designed for specific applications. Then, the filter design can be formulated as

$$\min_{h} ||\mathbf{s} - \mathbf{y}||_2^2 + \gamma f(\mathbf{F}, \mathbf{s}), \tag{4.47}$$

where $\mathbf{F}$ is the representing tensor of the hypergraph and $f(\cdot)$ is a penalty function designed for specific problems. For example, the total variation could be used as a penalty function for the purpose of smoothness. Other alternative penalty functions include the label rank, Laplacian regularization and spectrum. In Section 4.6, we shall provide some filter design examples.

4.6 Application Examples

In this section, we consider several application examples for our newly proposed HGSP framework. These examples illustrate the practical use of HGSP in some traditional tasks, such as filter design and efficient data representation. We also consider problems in data analysis, such as classification and clustering.

4.6.1 Data Compression

Efficient representation of signals is important in data analysis and signal processing. Among many applications, data compression attracts significant interests for efficient storage and transmission [100–102]. Projecting signals into a suitable orthonormal basis is a widely-used compression method [61]. Within the proposed HGSP framework, we propose a data compression method based on the hypergraph Fourier transform. We can represent N signals

Table 4.1: Compression Ration of Different Methods

size	16×16							256×256			
image	Radiation	People	load	inyang	stop	error	smile	lenna	mri	ct	AVG
IANH-HGSP	**1.52**	**1.45**	**1.42**	**1.47**	**1.52**	**1.39**	**1.40**	**1.57**	**1.53**	**1.41**	**1.47**
(α, β)-GSP	1.37	1.23	1.10	1.26	1.14	1.16	1.28	1.07	1.11	1.07	1.18
4 connected-GSP	1.01	1.02	1.01	1.01	1.04	1.02	1.07	1.04	1.05	1.07	1.03

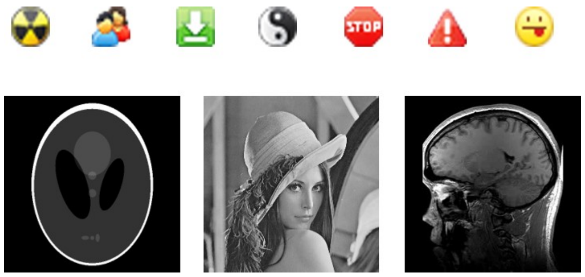


Figure 4.13: Test Set of Images.

in the original domain with C frequency coefficients in the hypergraph spectrum domain. More specifically, with the help of the sampling theory in Section 4.5, we can compress an K-bandlimited signal of N signal points losslessly with K spectrum coefficients.

To test the performance of our HGSP compression and demonstrate that hypergraphs may be a better representation of structured signals than normal graphs, we compare the results of image compression with those from GSP-based compression method [61]. We test over seven small size-16×16 icon images and three size-256×256 photo images, shown in Fig. 4.13.

The HGSP-based image compression method is described as follows. Given an image, we first model it as a hypergraph with the Image Adaptive Neighborhood Hypergraph (IANH) model [103]. To reduce complexity, we pick three closest neighbors in each hyperedge to construct a third-order adjacency tensor. Next, we can calculate the Fourier basis of the adjacency tensor as well as the bandwidth K of the hypergraph signals. Finally, we can represent the original images using C spectrum coefficients with $C = K$. For a large image, we may first cut it into smaller image blocks before applying HGSP compression to improve speed.

For the GSP-based method in [61], we represent the images as graphs with 1) the 4-

connected neighbor model [104], and 2) the distance-based model in which an edge exists only if the spatial distance is below α and the pixel distance is below β. The graph Fourier space and corresponding coefficients in the frequency domain are then calculated to represent the original image.

We use the compression ratio CR= N/C to measure the efficiency of different compression methods. A large CR implies higher compression efficiency. The result is summarized in Table 4.1, from which we can see that our HGSP-based compression method achieves higher efficiency than the GSP-based compression methods. Overall, hypergraph and HGSP lead to more efficient descriptions of structured data in most applications. With a more suitable hypergraph model and more developed methods, the HGSP framework could be a very new important tool in data compression.

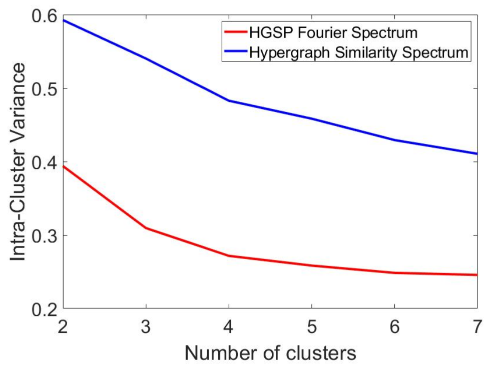
4.6.2 Spectral Clustering

Clustering problem is widely used in a variety of applications, such as social network analysis, computer vision, and communication problems. Among many methods, spectral clustering is an efficient clustering method [60, 106]. Modeling the dataset by a normal graph before clustering the data spectrally, significant improvement is possible in structured data[107]. However, such standard spectral clustering methods only exploit pairwise interactions. For applications where the interactions involve more than two nodes, hypergraph spectral clustering should be a more natural choice.

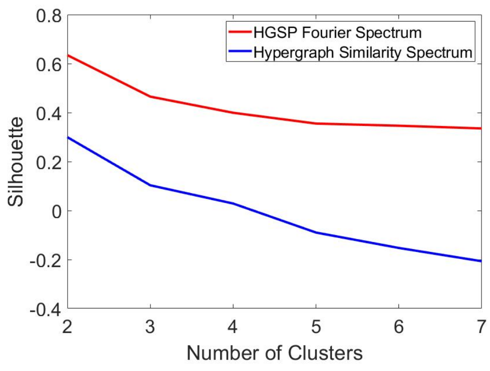
In hypergraph spectral clustering, one of the most important issues is how to define a suitable spectral space. In [24, 63], the authors introduced the hypergraph similarity spectrum for spectral clustering. Before spectral clustering, they first modeled the hypergraph structure into a graph-like similarity matrix. They then defined the hypergraph spectrum based on the eigenspace of the similarity matrix. However, since the modeling of hypergraph with a similarity matrix may result in certain loss of the inherent information, a more efficient spectral space defined directly over hypergraph is more desired as introduced in our

Algorithm 1 HGSP Fourier Spectral Clustering

```
1: **Input**: Dataset modeled in hypergraph $\mathcal{H}$, the number of clusters $k$.
2: Construct adjacency tensor $\mathbf{A}$ in Eq. (4.5) from the hypergraph $\mathcal{H}$.
3: Apply orthogonal decomposition to $\mathbf{A}$ and compute Fourier basis $\mathbf{f_i}$ together with Fourier
   frequency coefficient $\lambda_i$ using Eq. (4.14).
4: Find the first $E$ leading Fourier basis $\mathbf{f}_i$ with $\lambda_i \neq 0$ and construct a Fourier spectrum
   matrix $\mathbf{S} \in \mathbb{R}^{N \times E}$ with columns as the leading Fourier basis.
5: Cluster the rows of $\mathbf{S}$ into $k$ clusters using $k$-means clustering.
6: Put node $i$ in partition $j$ if the $i$-th row is assigned to the $j$-th cluster.
7: **Output**: $k$ partitions of the hypergraph dataset.
```



(a) Variance in the same cluster



(b) Average Silhouette in the hypergraph

Figure 4.14: Performance of Hypergraph Spectral Clustering.

HGSP framework. With HGSP, as the hypergraph Fourier space from the adjacency tensor has a similar form to the spectral space from adjacency matrix in GSP, we could develop the spectral clustering method based on the hypergraph Fourier space as in Algorithm 1.

To test the performance of the HGSP spectral clustering, we compare the achieved results with those from the hypergraph similarity method (HSC) in [63], using the zoo dataset [108]. To measure the performance, we compute the intra-cluster variance and the average Silhouette of nodes [109]. Since we expect the data points in the same cluster to be closer to each other, the performance is considered better if the intra-cluster variance is smaller. On the other hand, the Silhouette value is a measure of how similar an object is to its own cluster versus other clusters. A higher Silhouette value means that the clustering configuration is more appropriate.

The comparative results are shown in Fig. 4.14. Form the test result, we can see that our

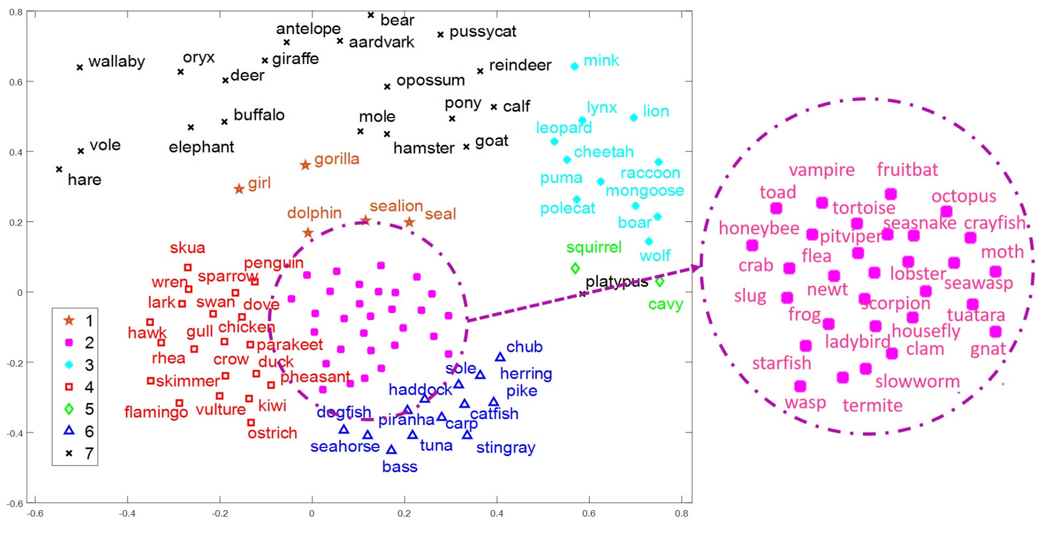


Figure 4.15: Cluster of Animals.

HGSP method generates a lower variance and a higher Silhouette value. More intuitively, we plot the clusters of animals in Fig. 4.15. Cluster 2 covers small animals like bugs and snakes. Cluster 3 covers carnivores whereas cluster 7 groups herbivores. Cluster 4 covers birds and Cluster 6 covers fish. Cluster 5 contains the rodents such as mice. One interesting category is cluster 1: although dolphins, sea-lions, and seals live in the sea, they are mammals and are clustered separately from cluster 6. From these results, we see that the HGSP spectral clustering method could achieve better performance and our definition of hypergraph spectrum may be more appropriate for spectral clustering in practice.

4.6.3 Classification

Classification problems are important in data analysis. Traditionally, these problems are studied by learning methods [110]. Here, we propose a HGSP-based method to solve the $\{\pm 1\}$ classification problem, where a hypergraph filter serves as a classifier.

The basic idea adopted for the classification filter design is label propagation (LP), where the main steps are to first construct a transmission matrix and then propagate the label

Algorithm 2 LP-HGSP Classification

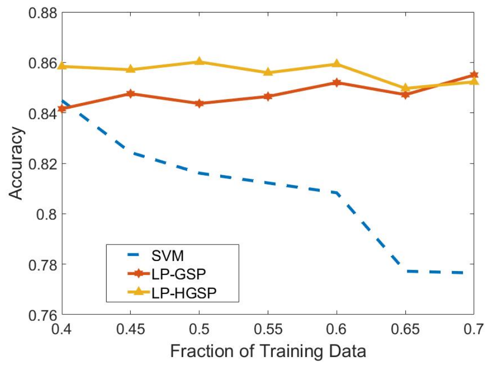
```
1: Input: Dataset s consisting of labeled training data and unlabeled test data.
2: Establish a hypergraph by similarities and set unlabeled data as $\{0\}$ in the signal $\mathbf{s}$.
3: Train the coefficients $\alpha$ of the LP-HGSP filter in Eq. (4.48) by minimizing the errors of
   signs of training data in $\mathbf{s}' = \mathbf{H}\mathbf{s}$.
4: Implement LP-HGSP filter. If ${\mathbf{s}'}_i > 0$, the $i$th data is labeled as 1; otherwise, it is labeled
   as $-1$.
5: Output: Labels of test signals.
```

based on the transmission matrix [111]. The label will converge after a sufficient number of shifting steps. Let $\mathbf{W}$ be the propagation matrix. Then the label could be determined by the distribution $\mathbf{s}' = \mathbf{W}^k\mathbf{s}$. We see that $\mathbf{s}'$ is in the form of filtered graph signal. Recall that in Section 4.5.2, the supporting matrix $\mathbf{P}$ has been shown to capture the properties of hypergraph shifting and total variation. Here, we propose a HGSP classifier based on the supporting matrix $\mathbf{P}$ defined in Eq. (4.41) to generate matrix

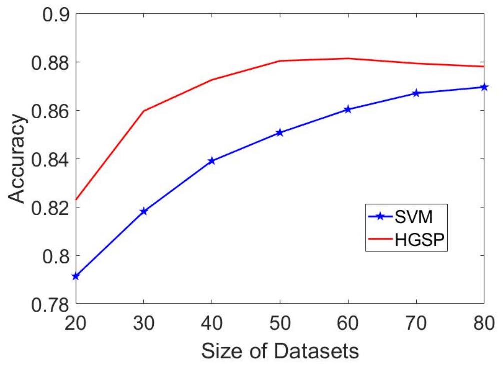
$$\mathbf{H} = (\mathbf{I} + \alpha_1\mathbf{P})(\mathbf{I} + \alpha_2\mathbf{P}) \cdots (\mathbf{I} + \alpha_k\mathbf{P}). \tag{4.48}$$

Our HGSP classifier is to simply rely on sign[$\mathbf{H}\mathbf{s}$]. The main steps of the propagated LP-HGSP classification method is described in Algorithm 2.

To test the performance of the hypergraph-based classifier, we implement them over the zoo datasets. We determine whether the animals have hair based on other features, formulated as a $\{\pm 1\}$ classification problem. We randomly pick different percentages of training data and leave the remaining data as the test set among the total 101 data points. We smooth the curve with 1000 combinations of randomly picked training sets. We compare the HGSP-based method against the SVM method with the RBF kernel and the label propagation GSP (LP-GSP) method [13]. In the experiment, we model the dataset as hypergraph or graph based on the distance of data. The threshold of determining the existence of edges is designed to ensure the absence of isolated nodes in the graph. For the label propagation method, we set $k = 15$. The result is shown in Fig. 4.16(a). From the result, we see that the label propagation HGSP method (LP-HGSP) is moderately better than LP-GSP. The



(a) Over datasets with a fixed datasize and different ratios of training data.



(b) Over datasets with different datasizes and a fixed ratio of training data.

Figure 4.16: Performance of Classifiers.

graph-based methods, i.e., LP-GSP and LP-HGSP, both perform better than SVM. The performance of SVM appears less satisfactory, likely because the dataset is rather small. Model-based graph and hypergraph methods are rather robust when applied to such small datasets. To illustrate this effect more clearly, we tested the SVM and hypergraph performance with new configurations by the increasing dataset size and the fixing ratio of training data in Fig. 4.16(b). In the experiment, we first pick different sizes of data subsets from the original zoo dataset randomly as the new datasets. Then, with each size of the new dataset, 40% data points are randomly picked as the training data, and the remaining data points are used as the test data. We average the results of 10000 times of experiments to smooth the curve. We can see from Fig. 4.16(b) that the performance of SVM shows significant improvement as the dataset size grows larger. This comparison indicates that SVM may require more data to achieve better performance, as shown in the comparative results of Fig. 4.16(a). Generally, the HGSP-based method exhibits better overall performance and shows significant advantages with small datasets. Although GSP and HGSP classifiers are both model-based, hypergraph-based ones usually perform better than graph-based ones, since hypergraphs provide a better description of the structured data in most applications.

4.6.4 Denoising

Signals collected in the real world often contain noises. Signal denoising is thus an important application in signal processing. Here, we design a hypergraph filter to implement signal denoising.

As mentioned in Section 4.3, the smoothness of a graph signal, which describes the variance of hypergraph signals, could be measured by the total variation. Assume that the original signal is smooth. We formulate signal denoising as an optimization problem. Suppose that $\mathbf{y} = \mathbf{s} + \mathbf{n}$ is a noisy signal with noise $\mathbf{n}$, and $\mathbf{s}' = h(\mathbf{F}, \mathbf{y})$ is the denoised data by the HGSP filter $h(\cdot)$. The denoising problem could be formulated as an optimization problem:

$$\min_h ||\mathbf{s}' - \mathbf{y}||_2^2 + \gamma \cdot ||\mathbf{s}' - \mathbf{s}'^{norm}_{<1>}||_2^2, \tag{4.49}$$

where the second term is the weighted quadratic total variation of the filtered signal $\mathbf{s}'$ based on the supporting matrix.

The denoising problem of Eq. (4.49) aims to smooth the signal based on the original noisy data $\mathbf{y}$. The first term keeps the denoised signal close to the original noisy signal, whereas the second term tries to smooth the recovered signal. Clearly, the optimized solution of filter design is

$$\mathbf{s}' = h(\mathbf{F}, \mathbf{y}) = [\mathbf{I} + \gamma(\mathbf{I} - \mathbf{P_s})^{\mathbf{T}}(\mathbf{I} - \mathbf{P_s})]^{-\mathbf{1}}\mathbf{y}, \tag{4.50}$$

where $\mathbf{P}_s = \sum_{i=1}^{N} \frac{\lambda_i}{\lambda_{\max}} \mathbf{f}_i \mathbf{f}_i^{\mathbf{T}}$ describes a hypergraph Fourier decomposition. From Eq. (4.50), we see that the solution is in the form of $\mathbf{s}' = \mathbf{H}\mathbf{y}$ for denoising, which adopts a hypergraph filter $h(\cdot)$ as

$$\mathbf{H} = [\mathbf{I} + \gamma(\mathbf{I} - \mathbf{P_s})^{\mathbf{T}}(\mathbf{I} - \mathbf{P_s})]^{-\mathbf{1}}. \tag{4.51}$$

The HGSP-based filter follows a similar idea to GSP-based denoising filter [70]. However, different definitions of the total variation and signal shifting result in different designs of HGSP vs. GSP filters. To test the performance, we compare our method with the basic

Table 4.2: MSE of Filtered Signal

γ	10e-5	10e-4	10e-3	10e-2	10e-1	1	10
Uniform Distribution: U(0, 0.1)							
GSP	0.0031	0.0031	0.0031	0.0026	**0.0017**	0.0895	0.4523
HGSP	0.0031	0.0031	0.0028	**0.0012**	0.0631	0.1876	0.4083
Wiener	0.0201						
Median	0.0142						
Normal Distribution: N(0, 0.09)							
GSP	0.790	0.790	0.0786	**0.0556**	0.0604	0.1286	0.4681
HGSP	0.0790	0.0585	**0.0305**	0.0778	0.1235	0.2374	0.4176
Wiener	0.0368						
Median	0.0359						
Normal Distribution: N(-0.02, 0.0001)							
GSP	**5.34e-04**	5.36e-04	5.54e-04	7.76e-04	0.0055	0.1113	0.4650
HGSP	**4.17e-04**	4.72e-04	4.86e-04	6.48e-04	0.0044	0.0868	0.3483
Wiener	0.0230						
Median	0.0096						

Wiener filter, Median filter, and GSP-based filter [70] using the image datasets of Fig. 4.13. We apply different types of noises. To quantify the filter performance, we use the mean square error (MSE) between each true signal and the corresponding signal after filtering. The results are given in Table 4.2. From these results, we can see that, for each type of noise and picking optimized γ for all the methods, our HGSP-based filter out-performs other filters.

4.6.5 Other Potential Applications

In addition to the application algorithms discussed above, there could be many other potential applications for HGSP. In this subsection, we suggest several potential applicable datasets and systems for HGSP.

- *IoT:* With the development of IoT techniques, the system structures become increasingly complex, which makes traditional graph-based tools inefficient to handle the high-dimensional interactions. On the other hand, the hypergraph-based HGSP is powerful in dealing with high-dimensional analysis in the IoT system: for example, data intelligence over sensor networks, where hypergraph-based analysis has already attracted significant attentions [112], and HGSP could be used to handle tasks like clustering, classification, and sampling.

- *Social Network:* Another promising application is the analysis of social network datasets. As discussed earlier, a hyperedge is an efficient representation for the multi-lateral relationship in social networks [62, 113]; HGSP can then be effective in analyzing multi-lateral node interactions.

- *Nature Language Processing:* Furthermore, natural language processing is an area that can benefit from HGSP. Modeling the sentence and language by hypergraphs [114,115], HGSP can be a tool for language classification and clustering tasks.

Overall, due to its systematic and structural approach, HGSP is expected to become an important tool in handling high-dimensional signal processing tasks that are traditionally addressed by DSP or GSP based methods.

4.7 Conclusions

In this chapter, we proposed a novel tensor-based framework of Hypergraph Signal Processing (HGSP) that generalizes the traditional GSP to high-dimensional hypergraphs. Our framework provided important definitions in HGSP, including hyerpgraph signals, hypergraph shifting, HGSP filters, frequency, and bandlimited signals. We presented basic HGSP concepts such as the sampling theory and filtering design. We show that hypergraph can serve as an efficient model for many complex datasets. We also illustrate multiple practical applications for HGSP in signal processing and data analysis, where we provided numerical results to validate the advantages and the practicality of the proposed HGSP framework. All the features of HGSP make it a powerful tool for IoT applications in the future.

Chapter 5

Hypergraph-based Multimedia Processing

5.1 Introduction

In this chapter, we investigate the use of hypergraph spectral analysis and introduce HGSP as tools in multimedia processing. Our contributions can be summarized as follows:

- We provide alternative definitions of hypergraph spectral operations within HGSP [123] for data analysis.
- We present guidelines for applying HGSP in image processing and introduce several applications, such as edge detection, compression, and video segmentation.
- We propose a spectrum-based hypergraph estimation for three-dimensional (3D) point clouds, together with some application examples in point cloud segmentation, resampling and denoising.

When presenting test results, we compare our proposed methods against benchmarks from traditional graph and learning methods. Our experiments demonstrate the effectiveness of HGSP and hypergraph spectral analysis in multimedia processing. Furthermore, we expect the proposed spectral operations to play important roles in the development of graph/hypergraph convolutional networks.

We organize this chapter as follows. We first provide some useful hypergraph operations for multimedia datasets in Section 5.2. Then, we investigate the applications of HGSP in image processing and point cloud processing in Section 5.3 and 5.4, respectively. Finally, we conclude our contributions in Section 5.5.

5.2 Hypergraph Operations for Multimedia Processing

In this section, we introduce important operations for the hypergraph frequency analysis useful in multimedia processing within the framework of HGSP aforementioned in Chapter 4.

5.2.1 Convolution

Convolution is an essential operation in traditional image processing. In DSP and GSP [126, 127], Fourier transform of convolution between two signals is equal to the product between their respective Fourier transforms. Similarly, we generalize the hypergraph convolution denoted by $\diamond$ as

$$\mathbf{x} \diamond \mathbf{y} = \mathcal{F}_C^{-1}(\mathcal{F}_C(\mathbf{x}) * \mathcal{F}_C(\mathbf{y})), \tag{5.1}$$

where $\mathcal{F}_C$ is the HGFT, $\mathcal{F}_C^{-1}$ is the iHGFT, and $*$ denotes Hadamard product [123]. This definition generalizes the property that the convolution in the vertex domain is equivalent to the product in the hypergraph spectral domain.

[3]Part of this chapter is reprinted, with permission, from [S. Zhang, S. Cui, and Z. Ding, "Hypergraph-based Image Processing", *2020 IEEE International Conference on Image Processing (ICIP)*, Abu Dhabi, UAE, 2020, pp. 216-220.].

5.2.2 Translation

The classic translation in DSP can be written as the convolution between the signal and an impulse function centered at a certain point. With the definition of hypergraph convolution, we define the hypergraph translation of an original signal $\mathbf{x} \in \mathbb{R}^N$ similar to that in GSP [127] as

$$\mathcal{T}_n \mathbf{x} = \sqrt{N} \mathbf{x} \diamond \Delta_n, \tag{5.2}$$

where the nth element of the Kronecker $\Delta_n \in \mathbb{R}^N$ is 1 and other elements are 0. Similar to translation in GSP, hypergraph translation is not like the time shift of signal in DSP. Instead, it represents a hypergraph convolution kernel localizing the information near the centered node $\mathbf{v}_n$ [117], which helps capture topological information among pixels in images.

5.2.3 Sampling and Interpolation

Sampling is an important operation in image processing, which selects a subset of individual data points to estimate the characteristics of the whole population. Similar to sampling signals in time and GSP [97], the HGSP sampling theory can be developed to sample signals over the vertex domain. Since the reduction of tensor order may break the structure of hypergraph and cannot always guarantee perfect recovery, we adopt the dimension reduction of each order. The sampling of a hypergraph signal is defined as follows:

Suppose that Q is the dimension of each sampled order. The sampling operation of a hypergraph signal $\mathbf{s}^{[\mathbf{M-1}]} \in \mathbb{R}^{\overbrace{N \times N \times \ldots \times N}^{M-1 \text{ times}}}$ is defined as

$$\mathbf{s}_{\mathbf{Q}}^{[\mathbf{M-1}]} = \mathbf{s}^{[\mathbf{M-1}]} \times_1 \mathbf{U} \times_2 \mathbf{U} \cdots \times_{M-1} \mathbf{U}, \tag{5.3}$$

where $\times_n$ denotes the n-mode product [123], the sampling operator is $\mathbf{U} \in \mathbb{R}^{Q \times N}$, and the sampled signal is $\mathbf{s}_{\mathbf{Q}}^{[\mathbf{M-1}]} \in \mathbb{R}^{\overbrace{Q\times Q\times\ldots\times Q}^{}_{M-1 \text{ times}}}$. The interpolation operation is defined by

$$\mathbf{s}^{[\mathbf{M-1}]} = \mathbf{s}_{\mathbf{Q}}^{[\mathbf{M-1}]} \times_1 \mathbf{T} \times_2 \mathbf{T} \cdots \times_{M-1} \mathbf{T}, \tag{5.4}$$

where the interpolation operator is $\mathbf{T} \in \mathbb{R}^{N\times Q}$.

Interested readers are referred to Section 4.5.1 for more properties of hypergraph-based sampling theory.

5.2.4 Hypergraph Stationary Process

Before providing details of the estimation, let us first introduce some new definitions and properties necessary for spectrum estimation.

Stationarity is a cornerstone property that facilities the analysis of random signals and observations in traditional signal processing [128]. It has equal importance in graph and hypergraph signal processing. Based on graph shifting introduced in [2], a definition of graph stationary process proposed in [128] can analyze the properties of the different observations of nodes, or the random signals over the graphs. Furthermore, [129] introduces a method to estimate the graph spectrum space and graph diffusion for multiple observations based on the graph stationary process. Similarly, the hypergraph stationary process can be defined to estimate hypergraph spectrum.

Now, let us introduce the definition of the hypergraph stationary process. In [123], a polynomial hypergraph filter based on supporting matrix is defined as

$$\mathbf{s}' = \sum_{k=1}^{a} \alpha_k \mathbf{P}^k \mathbf{s}, \tag{5.5}$$

where $\mathbf{P} = \lambda_{max}\mathbf{P_s}$ and the supporting matrix $\mathbf{P_s}$ is denoted by

$$\mathbf{P_s} = \frac{1}{\lambda_{\max}} \begin{bmatrix} \mathbf{f}_1 & \cdots & \mathbf{f}_N \end{bmatrix} \begin{bmatrix} \lambda_1 & & \\ & \ddots & \\ & & \lambda_N \end{bmatrix} \begin{bmatrix} \mathbf{f}_1^{\mathrm{T}} \\ \vdots \\ \mathbf{f}_N^{\mathrm{T}} \end{bmatrix} \tag{5.6}$$

to capture the overall spectral information of the hypergraph.

Similarly, based on the supporting matrix, a τ-step shifting operation is defined as $\mathbf{P}_\tau = \mathbf{P}^\tau$. Then, similar to the definition of the stationary process in traditional digital signal processing and graph signal processing, a strict-sense stationary process in HGSP can be defined as follows.

Definition 13. *(Strict-Sense Stationary Process) A stochastic signal $\mathbf{x} \in \mathbb{R}^N$ is strict-sense stationary over the hypergraph with $\mathbf{P}_\tau$ if and only if*

$$\mathbf{x} \overset{d}{=} \mathbf{P}_\tau \mathbf{x} \tag{5.7}$$

holds for any τ.

Since the strict-sense stationary is hard to achieve and analyze in the real datasets, we introduce the weak-sense stationary process similar to traditional digital signal processing.

Definition 14. *(Weak-Sense Stationary Process) A stochastic signal $\mathbf{x} \in \mathbb{R}^N$ is weak-sense stationary over the hypergraph with $\mathbf{P}_\tau$ if and only if*

$$\mathbb{E}[\mathbf{x}] = \mathbb{E}[\mathbf{P}_\tau \mathbf{x}] \tag{5.8}$$

and

$$\mathbb{E}[(\mathbf{P}_{\tau_1}\mathbf{x})((\mathbf{P}^H)_{\tau_2}\mathbf{x})^H] = \mathbb{E}[(\mathbf{P}_{\tau_1+\tau}\mathbf{x})((\mathbf{P}^H)_{\tau_2-\tau}\mathbf{x})^H] \tag{5.9}$$

hold for any τ, where $\mathbb{E}(\cdot)$ refers to the mean of observations and $(\cdot)^H$ is the Hermitian transpose.

From the definition of the weak-sense stationary process (WSS), Eq. (5.8) implies that the mean function of the signal must be constant, which is the same condition as in traditional digital signal processing (DSP) [130]. From the definition of supporting matrix, the (i, j)-th entry of $\mathbf{P}$ is the same as the (j, i)-th entry of $\mathbf{P}^H$, which indicates that $\mathbf{P}^H$ is the shifting in the opposite direction of $\mathbf{P}$. Then, the condition in Eq. (5.9) indicates that the hypergraph covariance function $K_{\mathbf{xx}}(\tau_1, -\tau_2) = K_{\mathbf{xx}}(\tau_1 + \tau, \tau - \tau_2) = K_{\mathbf{xx}}(\tau_1 + \tau_2, 0)$, which is also consistent with the definition in traditional DSP.

With the definition of the hypergraph stationary process, we have the following properties regarding the relationship between signals and hypergraph spectrum.

Theorem 5.1. *A stochastic signal* $\mathbf{x}$ *is WSS if and only if it has zero-mean and its covariance matrix has the same eigenvectors as the hypergraph spectrum basis, i.e.,*

$$\mathbb{E}[\mathbf{x}] = \mathbf{0} \tag{5.10}$$

and

$$\mathbb{E}[\mathbf{x}\mathbf{x}^H] = \mathbf{V}\Sigma_{\mathbf{x}}\mathbf{V}^H, \tag{5.11}$$

where $\mathbf{V} = [\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_N] \in \mathbb{R}^{N \times N}$ *are the hypergraph spectrum.*

The proof is provided in Appendix C.1. This theorem can be used to estimate the hypergraph spectrum, given multiple observations of several signal points.

5.3 Hypergraph-based Image Processing

5.3.1 Image Compression

Efficient compression of signals is important in data analysis and signal processing. Projecting signals onto a suitable orthonormal basis is very common in compression. Within the proposed HGSP framework, we can represent N signals in the original domain with C

frequency coefficients in the hypergraph spectrum domain. More specifically, according to the hypergraph sampling theory, we can losslessly compress a K-bandlimited signal of N signal points with K spectrum coefficients. The process is similar to the data compression we discussed in Section 4.6.1. Interested readers are referred to that part for more discussions and results.

5.3.2 Edge Detection

Convolution-based method is widely used in the edge detection of images in DSP. Similarly, hypergraph convolution-based operations can also play a role in image edge detection. Here, we introduce a hypergraph translation-based method for edge detection. To implement HGSP-based method, we first design a 9×9 mask and use it as a hypergraph with 81 nodes by the IANH model. Next, we implement hypergraph-based translation at the center of the mask. The summation of the translated result becomes the new value of the center node. Hovering the mask over the whole image, this process can be understood as blurring the hypergraph signals with the information of center node. We can design a threshold for the difference between the translated values and the original pixels to detect edges. We compared the proposed method with the Sobel and Prewitt methods [131] on three SIPI image datasets (http://sipi.usc.edu/database/) as shown in Fig. 5.1. From the result, we can see that HGSP-based method detects the details and the edges more explicitly. With deeper understanding of hypergraph convolution-based kernels, HGSP shows clear promise in edge detection.

5.3.3 Hypergraph Convolution Filter for Segmentation

Deep-learning methods, like graph and hypergraph convolutional networks [25,119,122] have achieved significant success in data analysis. However, these learning methods do not provide analysis and interpolation in the hypergraph spectrum domain. To explore hypergraph convolution operations, we propose a convolution-based filter for semi-supervised learning in

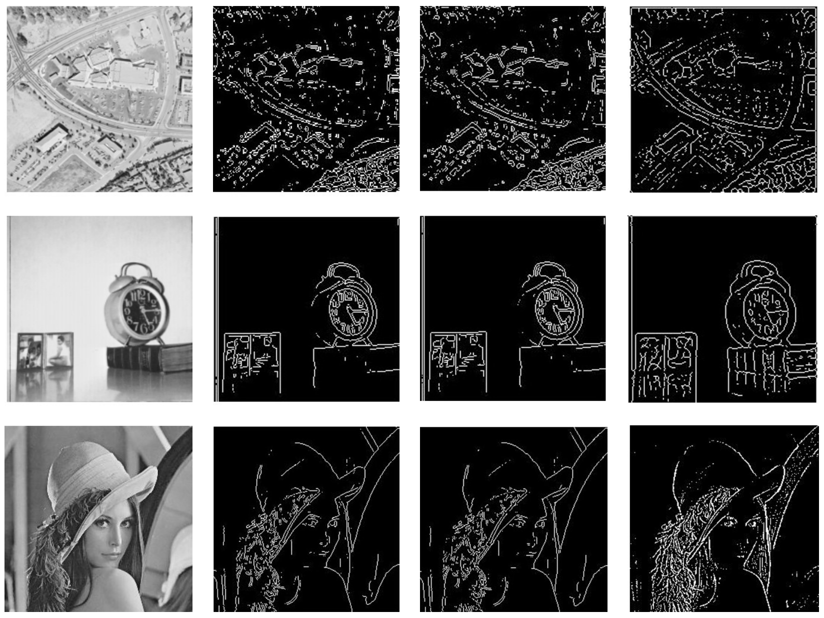


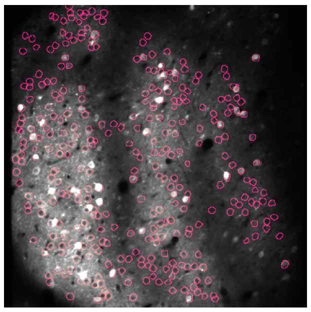
Figure 5.1: Results of Edge Detection.

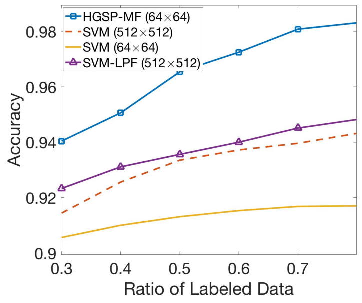
video segmentation. Define $\mathbf{V} = [\mathbf{f}_1 \quad \cdots \mathbf{f}_N]$. Given the definition of Eq. (5.1), a single-step convolution filter with parameter $\mathbf{y}$ on signal $\mathbf{x}$ is defined as

$$F_{\mathbf{y}}(\mathbf{x}) = \mathbf{x} \diamond \mathbf{y} = \mathbf{V} diag(\mathbf{f}_i^T \mathbf{y}) \mathbf{V}^T \mathbf{x}. \tag{5.12}$$

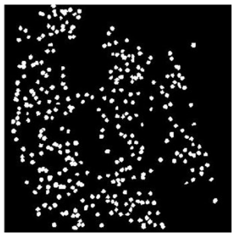
We now explain the use of convolution filter to segment. We consider a dataset in a biomedical application. In such datasets, a series of time-varying images are provided to record the activities of neuron shown as Fig. 5.2(a). The task here is to segment neurons from the background if we know part of labels in the ground truth shown as Fig. 5.2(c). Such a video with $K = 3024$ frames and $N = 512^2$ pixels in each frame can be modeled as a hypergraph with N nodes and each node has K observations.

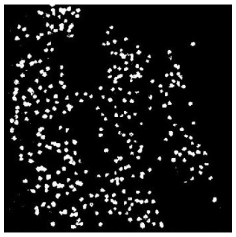
Assume that $\mathbf{x}_i \in \mathbb{R}^N, i = 1, \cdots K$ are the pixels in the ith frame. With the property of hypergraph stationary process in *Theorem 5.1*, we can easily estimate the hypergraph spectrum $\mathbf{V}$ for the time-varying images by applying decomposition on the normalized covariance matrix from K observations $\mathbf{x}_i$'s. Suppose that ℓ nodes have labels and $N - \ell$ do


(a) Detection of neurons



(b) Comparison with SVM.


(c) Ground Truth of Segmentation


(d) HGSP-based Segmentation with MF.

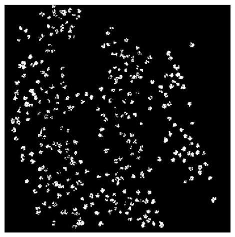

(e) SVM-based Segmentation with LPF.

Figure 5.2: Results of Video Segmentation.

not. We can estimate the filter parameters $\mathbf{y}$ by minimizing the error between the given labels $\mathbf{L} \in \mathbb{R}^{\ell}$ and the ℓ corresponding filtered signals $F_{\mathbf{y}}(\mathbf{x})_{\ell} \in \mathbb{R}^{\ell}$, i.e.,

$$\min_{\mathbf{y}} ||\mathbf{L} - F_{\mathbf{y}}(\tilde{\mathbf{x}})_l||_2^2, \tag{5.13}$$

where $\tilde{\mathbf{x}} = \frac{1}{K}\sum \mathbf{x}_i$ is the pixel average. More specifically, this optimization problem has a solution at

$$\mathbf{y} = \mathbf{W}^{-1}\mathbf{L}, \tag{5.14}$$

where each row of $\mathbf{W} \in \mathbb{R}^{l\times N}$ is the corresponding row of $\mathbf{V}diag(\mathbf{f}_i^T\tilde{\mathbf{x}})\mathbf{V}^T$ with the same index as the labeled data. From the estimated parameters $\mathbf{y}$, we obtain the filtered signals for all the nodes in $F_{\mathbf{y}}(\tilde{\mathbf{x}})$ and apply a threshold to determine its label (i.e., $\{\pm 1\}$) in the binary classification. To tackle possible noise amplification, we further apply a 4×4 median filter (MF) after the convolution filter.

In the experiment, we cut the original 512×512 images into 64×64 non-overlapping blocks

Table 5.1: Comparison of Computation Time (in Seconds)

HGSP(64×64)	SVM (64×64)	SVM (512×512)
2413	403	82813

to lower complexity. One result with $\ell = 40\% \cdot N$ labeled data achieves 95.06% accuracy, as shown in Fig. 5.2(d) and shows much clearer segmentation compared to the SVM result after a 4×4 averaging-kernel lowpass filter (LPF) [132] in Fig. 5.2(e). Fig. 5.2(b) further compares our method to SVM with and without LPF in terms of estimated label accuracy. From the comparison, we can see that the performances of SVM improves with larger processing blocks at the cost of computation complexity. LPF can improve the performance of SVM results. Still, the HGSP-based method achieves the best performance. We measured the computation time for these methods in Table 5.1. It is clear that HGSP-based methods achieves superior accuracy with moderate computation cost.

5.4 Hypergraph-based Point Cloud Processing

5.4.1 Motivation

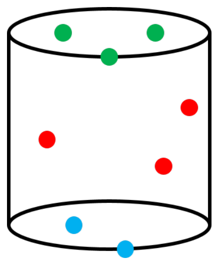
Recent developments in depth sensors and softwares make it easier to capture the features and create a three-dimensional (3D) model for an object and its surroundings [133]. In particular, with low-cost scanners such as light detection and ranging (LIDAR) and Kinect, a new data structure known as the point cloud has achieved significant successes in many areas, including virtual reality, geographic information system, reconstruction of art document and high-precision 3D maps for self-driving cars [134]. A point cloud consists of 3D coordinates with attributes such as color, temperature, texture, and depth [135]. Owing to the easy access to scanning sensors and the huge need in describing the 3D features, the use of point clouds has attracted significant attentions in areas of computer vision, virtual reality, and

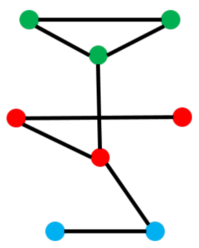
[4]Part of this chapter is reprinted, with permission, from [S. Zhang, S. Cui, and Z. Ding, "Hypergraph Spectrum Analysis and Processing in 3D Point Clouds," in *IEEE Transactions on Image Processing*, vol. 30, pp. 1193-1206, Dec. 2021.].

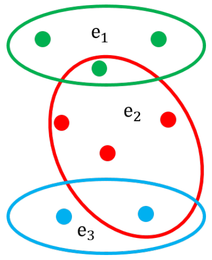
medical science. How to process the point clouds efficiently becomes an important topic of research in many 3D imaging and vision systems.

To analyze the features of point cloud, the first step is to construct an analytical model to represent the 3D structures. The literature provides several different models. In [136], the 3D space is partitioned into several boxes or voxels, and the point clouds are then discretized therein. One disadvantage of voxels is that a dense grid is required to achieve fine resolution, leading to spatial inefficiency [135]. A spatially efficient approach [137, 138] is the octree representation of point clouds. An octree is a tree data structure in which each node has exactly eight children. It can partition a 3D space recursively, and represent the point clouds with partitioned boxes. Although efficient, octree suffers from discretization errors [135]. The bd-tree is another spatial decomposition technique and is robust in highly cluttered point cloud dataset. However, compared to octree structures, bd-trees are more difficult to update.

Recently, graphs and graph signal processing (GSP) have found applications in modeling point clouds. For example, the authors of [135] construct a graph based on pairwise point distances. Some other works, such as [139, 140], construct graphs based on the k-nearest neighbors, where each vertex (point) has an edge connection to its k nearest neighbors. There are several clear connections between graph features and point cloud characteristics. For example, the smoothness over a graph can describe the flatness of surfaces in point clouds. GSP-based tools such as filters and graph learning methods can process the point clouds and have shown great success because of the graph model's ability to capture the underlying geometric structures. However, graph-based methods still face some challenges, such as limited orders and measurement inefficiency. In a traditional graph, each edge can only connect two nodes, constraining graph-based models to describe only pairwise relationships. However, a multilateral relationship among multiple nodes is far more informative in a point cloud model. For example, points (i.e., nodes) on the same surface of a point cloud exhibit a strong multilateral relationship, which cannot be easily captured by an edge of a traditional


(a) A 3D Shape with Eight Nodes: Green in the Top, Red in the Side, Blue in the Bottom.


(b) Distance-based Graph Model with Eight Edges.



(c) Hypergraph Model with Three Hyperedges.

Figure 5.3: Examples of Geometric Models of Point Clouds.

graph. In fact, construction of an efficient graph for a given dataset is always an open question. Thus, studies on point clouds can benefit from more general and efficient models.

To develop an efficient model for point clouds, we explore a high-dimensional graph model, known as hypergraph [123]. Hypergraph can be a useful model in processing 3D point clouds. A hypergraph $\mathcal{H} = \{\mathcal{V}, \mathcal{E}\}$ consists of a set of nodes $\mathcal{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_K\}$ and a set of hyperedges $\mathcal{E} = \{\mathbf{e}_1, \ldots, \mathbf{e}_K\}$. Each hyperedge in a hypergraph can connect more than two nodes. Obviously, a normal graph is a special case of a hypergraph, where each hyperedge degrades to connect two nodes exactly. The hyperedge in a hypergraph can characterize the multilateral relationship among several related nodes (e.g., on a surface), thereby making hypergraph a natural and intuitive model for point clouds. For example, a 3D shape together with its geometric models are shown as Fig. 5.3. Since each edge only connects two nodes as nodes are treated equivalently in a traditional graph as Fig. 5.3(b), it is hard to distinguish from the graph structure which surface a node belongs to. However, from the hypergraph model in Fig. 5.3(c), we can easily identify surfaces of nodes according to high-dimensional hyperedges. Furthermore, advances in hypergraph signal processing (HGSP) [123] are providing more hypergraph tools to process high-dimensional cross-features and multilateral interactions among nodes, such as HGSP-based filters and spectral analysis, for effective point cloud processing.

However, processing the point clouds based on hypergraphs still poses several challenges. Similar to GSP, the first problem lies in the construction of hypergraph for point clouds. The traditional hypergraph construction method for a general dataset relies on data structure. For example, in [114], a hypergraph model is constructed according to the sentence structure in natural language processing. The k-nearest neighbor model is another method to construct the hypergraph. In [123], a hypergraph can be formed from the feature distances for an animal dataset to achieve clustering. However, such distance-based or structure-based model may be rather lossy in information preservation. For example, the structure-based method may not preserve the correlation of some irregular structures, whereas the k-nearest neighbor method may narrowly emphasize the distance information. In addition to hypergraph construction, another issue in analyzing point cloud with hypergraph tools is the computation complexity of the spectrum space. In the HGSP framework, spectrum-based analysis plays an important role but needs to compute the spectral space. Usually, the computation of hypergraph spectrum is based on orthogonal-CP decomposition [36], which incurs high-complexity when there are many nodes. Another challenge in point cloud processing is the effect of noise and outliers. Since a hypergraph model is constructed from observed data, noise can distort the hypergraph and degrade the performances of HGSP. Thus, mitigating noise effect and robustly estimating the hypergraph model for point clouds pose a significant challenge.

This section addresses the aforementioned problems. We propose novel spectrum-based hypergraph construction methods for both clean and noisy point clouds. For clean point clouds, we first estimate their spectrum components based on the hypergraph stationary process and optimally determine their frequency coefficients based on smoothness to recover the original hypergraph structure. For noisy point clouds, we introduce a method for joint hypergraph structure estimation and data denoising. We shall illustrate the effectiveness of the proposed hypergraph construction and spectrum estimation in two point cloud applications: sampling and denoising. Our experimental results clearly establish a connection between hypergraph frequencies and point cloud features. The performance improvement in both applications demonstrates the strength and power of hypergraph in point cloud processing and the practical value of our estimation methods.

5.4.2 Hypergraph Spectrum Estimation for Clean Point Clouds

A point cloud is a set of 3D points obtained from sensors or generated synthetically, where each point is attributed with coordinates and other features, such as color [141]. Since the 3D coordinates are basic features of a point cloud, in this chapter, we primarily focus on point clouds characterized by their coordinates. We consider a matrix representation of the point clouds, where a point cloud with N nodes is denoted by a location matrix

$$\mathbf{s} = [\mathbf{X_1} \quad \mathbf{X_2} \quad \mathbf{X_3}] = \begin{bmatrix} \mathbf{s}_1^T \\ \mathbf{s}_2^T \\ \ddots \\ \mathbf{s}_N^T \end{bmatrix} \in \mathbb{R}^{N \times 3}, \tag{5.15}$$

where $\mathbf{X}_i$ denotes a vector of the ith coordinates of all the points, and $\mathbf{s}_i$ is the three coordinates of ith point. With the information of coordinates, different models, such as graphs [135] and octrees [137], can be constructed to analyze the point clouds.

To process the 3D point clouds within the HGSP framework in Section 4.3, the first step is to construct an optimal hypergraph to model the point clouds. As we mentioned in the Section 5.4.1, it is time-comsuming and inefficient to first construct a hypergraph structure before tensor decomposition to obtain the hypergraph spectrum. Instead, we propose to directly estimate the hypergraph spectral pairs based on the observed data, and then recover the original representing tensor with Eq. (4.14). In this section, we first estimate the hypergraph spectrum components $\mathbf{f}_r$'s based on the hypergraph stationary process, and optimize the frequency coefficients λ_r's based on the smoothness for original point clouds.

Algorithm 3 Estimation of Hypergraph Spectrum

```
1: Input: Point cloud dataset $\mathbf{s} = [\mathbf{X_1} \quad \mathbf{X_2} \quad \mathbf{X_3}] \in \mathbb{R}^{N\times 3}$.
2: Calculate the mean of each row in $\mathbf{s}$, i.e.,
   $\bar{\mathbf{s}} = (\mathbf{X_1} + \mathbf{X_2} + \mathbf{X_3})/3$;
3: Normalize the original point cloud data as zero-mean in each row, i.e., $\mathbf{s}' = [\mathbf{X_1} - \bar{\mathbf{s}}, \mathbf{X_2} - \bar{\mathbf{s}}, \mathbf{X_3} - \bar{\mathbf{s}}]$;
4: Calculate the eigenvectors $\{\mathbf{f}_1, \cdots, \mathbf{f}_N\}$ for $R_{\mathbf{s}'} = \mathbf{s}'(\mathbf{s}'^T)$ via SVD;
5: Output: Hypergraph spectrum $\mathbf{V} = [\mathbf{f}_1, \cdots, \mathbf{f}_N]$.
```

Estimation of Hypergraph Spectrum Components

In this part, we propose a method to estimate the hypergraph spectral components based on the hypergraph stationary process.

The three coordinates of a point can be interpreted as three observations of the point from different angles, which describe the underlying multilateral relationship. Thus, we can assume that the point cloud signals follow the stationary process over the estimated underlying hypergraph structure. If the point cloud signals $\mathbf{s}$ follow the hypergraph stationarity, it should satisfy Eq. (5.10) and Eq. (5.11). Thus, a spectrum estimation method can be based on hypergraph stationarity. The details of the algorithm is described as follows in Algorithm 3.

With *Theorem 5.1*, we can directly obtain an estimation of the hypergraph spectrum based on the hypergraph stationarity. Note that, here, we assume all the observations are from a clean point cloud without noise. The case of noisy point clouds will be discussed later.

Different from the traditional column-wise mean of the coordinates in Eq. (5.15), we use a row-wise mean of the coordinates to calculate the spectrum. Traditional column-wise mean methods [142] is based on principal component analysis (PCA) in a local region for dimension reduction and visualization of the data structure with a covariance matrix in $\mathbb{R}^{3\times 3}$. Typical applications of such PCA-based methods include points classification and region growing in segmentation [143, 144]. However, our objective is different. Since our goal is to estimate a hypergraph spectrum matrix in $\mathbb{R}^{N\times N}$ based on hypergraph stationary process rather than to reduce the data structure dimensions, the three coordinates represent three observations (or hypergraph signals) of the N nodes. Consequently, we apply the row-wise mean instead, unlike column-wise means used in the PCA-based methods.

Estimation of Frequency Coefficients

Next, we discuss how we estimate the hypergraph frequency coefficients with the spectrum components based on the hypergraph smoothness.

In real applications, the large-scale networks are usually sparse, which makes it meaningful to infer that most entries of the hypergraph representing tensor for real datasets are zero [145]. In addition, the smoothness of signals is a widely-used assumption when estimating the underlying structure of graphs and hypergraphs [146]. Thus, the estimation of the hypergraph representing tensor with known spectrum components for a given dataset $\mathbf{s}$ can be generally formulated as

$$\min_{\boldsymbol{\lambda}} \quad \alpha \text{Smooth}(\mathbf{s}, \boldsymbol{\lambda}, \mathbf{f}_r) + \beta ||\mathbf{A}||_T^2 \tag{5.16}$$

$$s.t. \quad \mathbf{A} = \sum_{r=1}^{N} \lambda_r \cdot \underbrace{\mathbf{f}_r \circ \ldots \circ \mathbf{f}_r}_{\text{M times}}. \tag{5.17}$$

$$\mathbf{A} \in \mathcal{A}. \tag{5.18}$$

$$||\mathbf{A}||_T = \sqrt{\sum_{i_1, i_2, \cdots, i_M = 1}^{N} a_{i_1 i_2 \cdots i_M}^2}. \tag{5.19}$$

The constraint set $\mathcal{A}$ in (5.18) includes the prior information of the representing tensor. For example, if the representing tensor is the adjacency tensor, its entries should be non-negative. In the constraint of (5.19), $||\mathbf{A}||_T$ is the tensor norm which controls the sparsity of the hypergraph structure. Here, we consider the Frobenius-like norm [27] since we aim to build a connection between spectra and hypergraph structures. The use of other tensor norms hypergraph applications can be exploited in future works. The smoothness function Smooth$(\mathbf{s}, \boldsymbol{\lambda}, \mathbf{f}_r)$ can be designed for specific problems. Typical functions can be hypergraph

Laplacian regularization, label ranking, and total variation [123].

Here, the HGSP-based total variation is used to measure the smoothness of signals over estimated hypergraph. In Eq. (4.24a), we have

$$\mathbf{A}\mathbf{f}_r^{[M-1]} = \sum_{i=1}^{N} \lambda_i \mathbf{f}_i (\mathbf{f}_i^{\mathrm{T}} \mathbf{f}_r)^{M-1} = \lambda_r \mathbf{f}_r. \tag{5.20}$$

With the supporting matrix defined in Eq. (5.6), we also have

$$\mathbf{P_s}\mathbf{f}_r = \frac{1}{\lambda_{max}} \sum_{i=1}^{N} \lambda_i \mathbf{f}_i (\mathbf{f}_i^{\mathrm{T}} \mathbf{f}_r) = \frac{\lambda_r}{\lambda_{max}} \mathbf{f}_r. \tag{5.21}$$

Consequently, the total variation can be written with the supporting matrix as follows:

$$\mathbf{TV}(\mathbf{s}) = ||\mathbf{s} - \mathbf{P_s}\mathbf{s}||_k, \tag{5.22}$$

where k ls application dependent. More details can be found in Section 4.3.5.

For convenience, we use the quadratic-form total variation based on the supporting matrix to describe the hypergraph smoothness, i.e.,

$$\mathbf{TV}(\mathbf{s}) = ||\mathbf{s} - (1/\lambda_{max})\mathbf{P}\mathbf{s}||_2^2. \tag{5.23}$$

This form of smoothness function suggested in [123] can capture the differences between one node and its neighbors over hypergraph. Since the signals are smooth over the estimated hypergraph, observations are also smooth. Thus, the final smoothness function for point cloud $\mathbf{s} = [\mathbf{X_1} \quad \mathbf{X_2} \quad \mathbf{X_3}]$ is

$$\begin{aligned}\text{Smooth}(\mathbf{s}, \boldsymbol{\lambda}, \mathbf{f}_r) &= \sum_{i=1}^{3} ||\mathbf{X}_i - \mathbf{P_s}\mathbf{X}_i||_2^2 \\ &= \sum_{i=1}^{3} ||\mathbf{X}_i - \sum_r \sigma_r(\mathbf{f}_r^T\mathbf{X}_i)\mathbf{f}_r||_2^2 \\ &= \sum_{i=1}^{3} ||\mathbf{X}_i - \mathbf{W}_i\boldsymbol{\sigma}||_2^2, \qquad (5.24)\end{aligned}$$

where $\mathbf{W}_i = [(\mathbf{f}_1^T\mathbf{X}_i)\mathbf{f}_1 \quad (\mathbf{f}_2^T\mathbf{X}_i)\mathbf{f}_2 \quad \cdots \quad (\mathbf{f}_N^T\mathbf{X}_i)\mathbf{f}_N]$, $\sigma_r = \lambda_r/\lambda_{max}$ and $\boldsymbol{\sigma} = [\sigma_1 \cdots \sigma_N]^T$.

Moreover, the tensor norm of a given hypergraph has the following property with the frequency coefficients.

Theorem 5.2. *Given a representing tensor* $\mathbf{A} = \sum_{r=1}^{N} \lambda_r \cdot \underbrace{\mathbf{f}_r \circ \ldots \circ \mathbf{f}_r}_{M\ times}$*, the tensor norm* $||\mathbf{A}||_T^2 = \sum_{i_1,i_2,\cdots,i_M=1}^{N} a_{i_1i_2\cdots i_M}^2$ *can be written in the form of frequency coefficients as*

$$||\mathbf{A}||_T^2 = \sum_{r=1}^{N} \lambda_r^2 = \boldsymbol{\lambda}^T\boldsymbol{\lambda}, \qquad (5.25)$$

where $\boldsymbol{\lambda} = [\lambda_1 \quad \lambda_2 \quad \cdots \quad \lambda_N]^T$.

The proof is in the Appendix C.2. This property can help us build a connection from the tensor norm to the frequency coefficients directly.

Now, if we consider the representing tensor as the adjacency tensor and each hyperedge consists of three nodes since at least three nodes are required to construct a surface, we optimize the normalized frequency coefficients $\boldsymbol{\sigma} = \frac{1}{\lambda_{max}}\boldsymbol{\lambda} = [\sigma_1 \quad \sigma_2 \quad \cdots \quad \sigma_N]^T$ via

$$\min_{\boldsymbol{\sigma}} \quad \alpha \sum_{i=1}^{3} ||\mathbf{X}_i - \mathbf{W}_i\boldsymbol{\sigma}||_2^2 + \beta\boldsymbol{\sigma}^T\boldsymbol{\sigma} \qquad (5.26)$$

$$s.\,t. \quad 0 \le \sigma_r \le \max_i \sigma_i = 1, \qquad (5.27)$$

$$\sum_{r=1}^{N} \sigma_r f_{r,i_1} f_{r,i_2} f_{r,i_3} \ge 0, \quad i_1, i_2, i_3 = 1, 2, \cdots, N. \qquad (5.28)$$

Algorithm 4 Estimation of Frequency Coefficient

```
1: Input: Point cloud dataset $\mathbf{s} = [\mathbf{X_1}, \mathbf{X_2}, \mathbf{X_3}] \in \mathbb{R}^{N\times 3}$, hypergraph spectrum $\mathbf{V} = [\mathbf{f}_1, \cdots, \mathbf{f}_N]$.
2: for i=1,2,...,iter do:
3:     Set $\sigma_i = 1$ as the maximal normalized eigenvalue.
4:     Solve the optimization problem in Eq. (5.26).
5: end for
6: Find the optimal $i$ to minimize the target function.
7: The optimal coefficients $\boldsymbol{\sigma}$ is the solution of Eq. (5.26) correlated to the optimal $i$.
8: Output: Frequency coefficients $\boldsymbol{\sigma}$.
```

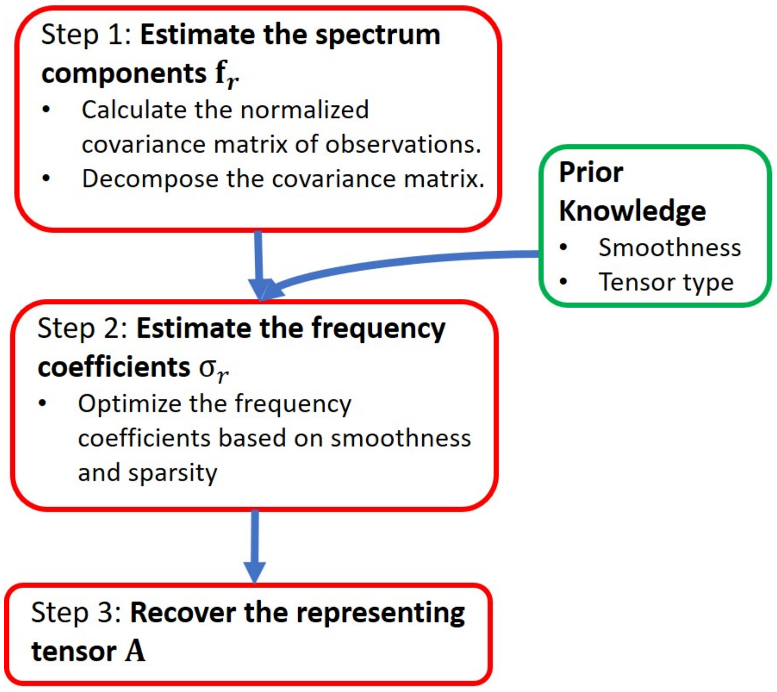

Step 1: Estimate the spectrum components $\mathbf{f}_r$
• Calculate the normalized covariance matrix of observations.
• Decompose the covariance matrix.
Prior Knowledge
• Smoothness
• Tensor type
Step 2: Estimate the frequency coefficients σ_r
• Optimize the frequency coefficients based on smoothness and sparsity
Step 3: Recover the representing tensor $\mathbf{A}$

Figure 5.4: Estimation of Hypergraph Spectral Pairs for Original Point Clouds

The constraint (5.28) limits the estimated representing tensor as the adjacency tensor. The constraint (5.27) is the nonnegative constraint on weight and the factor [36]. Clearly, the optimization is non-convex with the constraint $\max_i \sigma_i = 1$. However, if the position of the maximal frequency is known, the optimization problem can be solved by tools such as cvx [147, 148]. Thus, we can develop the following algorithm to estimate the frequency coefficients in Algorithm 4.

Note that, since we consider clean point cloud without noise, we usually set parameter $\alpha \leq \beta$. Then, from the estimated spectrum pair $(\mathbf{f}_r, \sigma_r)$ under normalization, we can recover the original adjacency tensor as Eq. (5.17). Hence, the hypergraph construction process for a clean point cloud can be summarized as Fig. 5.4. The recovery of original adjacency tensor is not always necessary in practical applications since storing the representing tensor is less efficient than storing the spectrum pairs.

5.4.3 Joint Spectrum Estimation and Denoising

In practical 3D imaging, perturbations such as noises and outliers often exist when generating a point cloud of an unknown object. These noises may significantly affect the performance of point cloud processing since many existing algorithms require quality datasets. Thus, denoising remains a vital issue in practical point cloud applications.

Usually, to denoise point sets with sharp features is difficult, especially when the noise is large, as such features are hard to distinguish from noise effect. Generally, smoothness-based methods are common. In [150], a method based on L_0 norm of differences between k-nearest neighbors is introduced. In [149], Laplacian regularization is used to describe smoothness and to denoise noisy point sets. Other works, such as [139,151], minimize the total variation over graphs to denoise the point sets. Although smoothness-based methods have achieved notable successes, how to interpret and define an effective smoothness function for a general point set remains open. Furthermore, for graph-based smoothness methods, the construction of graph model remains a critical problem, since traditional methods based on distance suffers from the imprecise location measurement. To this end, a more general definition of smoothness and a more efficient denoising method for arbitrary point clouds are highly desirable.

In this subsection, we introduce a joint method to simultaneously estimate the hypergraph structure and denoise noisy point clouds. In Section 5.4.2, we already introduce an estimation method of spectral pair $(\mathbf{f}_r, \sigma_r)$ for clean point clouds. A similar construction process can be developed for the noisy point clouds. As the estimation of spectrum components only depends on the observed data, we need to denoise the noisy observations while optimizing the frequency coefficients. As already discussed, the problem of denoising a signal on a hypergraph can be written as a convex minimization problem with the constraints that denoised signals should be smooth over the hypergraph. Accordingly, the general process of hypergraph denoising and estimation can be summarized as the following steps:

- Step 1: Estimate the approximated hypergraph spectrum components from the observed noisy point clouds;
- Step 2: Jointly estimate frequency coefficients and denoise the noisy observations;
- Step 3: Update the noisy observations as denoised data and repeat Step 1 until enough iterations.

To estimate hypergraph spectral components of noisy data, the process is the same as Algorithm 3 based on hypergraph stationary process. To jointly estimate the frequency coefficients to recover the original underlying structure and to denoise the noisy point clouds, we propose the following objective. Given N noisy points $\mathbf{s} = [\mathbf{X_1} \quad \mathbf{X_2} \quad \mathbf{X_3}]$, the joint estimation task can be formulated as

$$\begin{aligned} \min_{\boldsymbol{\sigma},\mathbf{Y}} \quad & \sum_{i=1}^{3} [||\mathbf{X}_i - \mathbf{Y}_i||_2^2 + \alpha||\mathbf{X}_i - \mathbf{W}_i\boldsymbol{\sigma}||_2^2] + \beta||\mathbf{A}||_T^2 \\ s.t. \quad & \mathbf{A} = \sum_{r=1}^{N} \lambda_r \cdot \underbrace{\mathbf{f}_r \circ \ldots \circ \mathbf{f}_r}_{\text{M times}} \in \mathcal{A}, \\ & 0 \le \sigma_r \le \max_i \sigma_i = 1, \\ & \mathbf{W}_i = [(\mathbf{f}_1^T\mathbf{X}_i)\mathbf{f}_1 \quad (\mathbf{f}_2^T\mathbf{X}_i)\mathbf{f}_2 \quad \cdots \quad (\mathbf{f}_N^T\mathbf{X}_i)\mathbf{f}_N]. \end{aligned} \tag{5.29}$$

The resulting $\mathbf{Y} = [\mathbf{Y_1} \quad \mathbf{Y_2} \quad \mathbf{Y_3}]$ is the denoised point clouds, and (α, β) are two positive regularization parameters. The first part in Eq. (5.29) lets the denoised point cloud maintain the observed structural features. The second part is the smoothness function derived from Eq. (5.24) which adjusts positions of noisy points. The third part is the tensor norm regularization to control hypergraph sparsity.

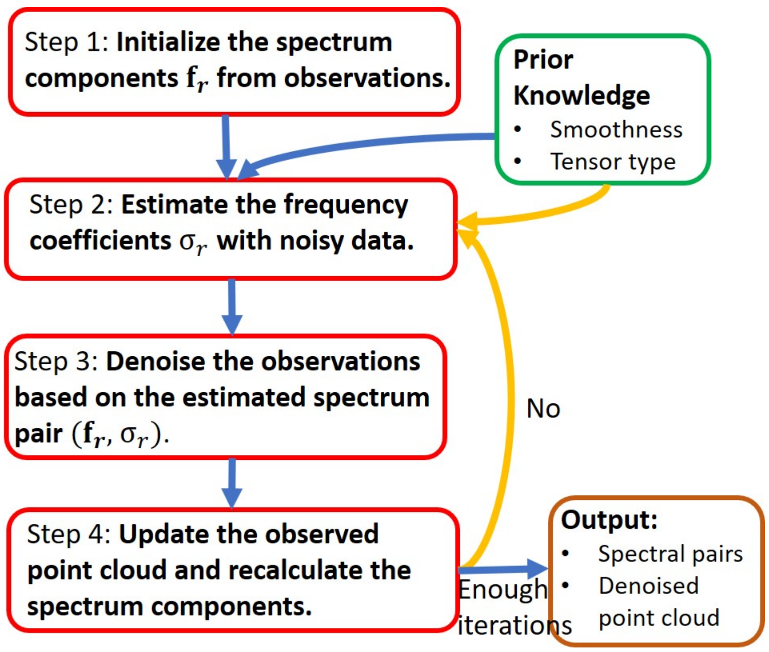

Step 1: Initialize the spectrum components $\mathbf{f}_r$ from observations.
Prior Knowledge
• Smoothness
• Tensor type
Step 2: Estimate the frequency coefficients σ_r with noisy data.
Step 3: Denoise the observations based on the estimated spectrum pair $(\mathbf{f}_r, \sigma_r)$.
No
Step 4: Update the observed point cloud and recalculate the spectrum components.
Enough iterations
Output:
• Spectral pairs
• Denoised point cloud

Figure 5.5: Joint Hypergraph Estimation and Denoising for Noisy Point Cloud.

The optimization problem of Eq. (5.29) is not convex in $\mathbf{Y}$ and $\boldsymbol{\sigma}$. Therefore, similar to [146], we split the problem into two subproblems. For each subproblem, we fix one variable set to solve the other one. Upon convergence, the solution corresponds to a local minimum and not necessarily a global minimum.

We first initialize $\mathbf{Y}$ as the observed signals $\mathbf{X}$ and solve the following problem similar to that in Section 5.4.2.

$$\min_{\boldsymbol{\sigma}} \alpha \sum_{i=1}^{3} ||\mathbf{X}_i - \mathbf{W}_i\boldsymbol{\sigma}||_2^2 + \beta\boldsymbol{\sigma}^T\boldsymbol{\sigma} \tag{5.30}$$

$$\begin{aligned} s.t. \quad & 0 \leq \sigma_r \leq \max_i \sigma_i = 1, \\ & \sum_{r=1}^{N} \sigma_r f_{r,i_1} f_{r,i_2} f_{r,i_3} \geq 0, \quad i_1, i_2, i_3 = 1, 2, \cdots, N. \end{aligned}$$

This problem can be solved similarly to the solution of clean point cloud with Algorithm 4.

Once the estimated frequency coefficients are found, we solve the subproblem of point

Algorithm 5 Joint Hypergraph Estimation and Point Cloud Denoising

```
1: Input: Noisy observations of point clouds $\mathbf{s} = [\mathbf{X_1}, \mathbf{X_2}, \mathbf{X_3}] \in \mathbb{R}^{N\times 3}$.
2: Initialization: Calculate the spectrum components $\mathbf{f}_r$'s from the observed point cloud
   $\mathbf{s}$ as Algorithm 3.
3: for i=1,2,...,iter do:
4:     Find the optimal $\boldsymbol{\sigma}$ for the first subproblem in Eq. (5.30) with Algorithm 4.
5:     Solve the optimization problem in Eq. (5.31) with $\mathbf{Y}$ in Eq. (5.32).
6:     Update the observed signals as $\mathbf{Y}$ and recalculate the spectrum components $\mathbf{f}_r$'s.
7: end for
8: Output: Spectral pairs $(\mathbf{f}_r, \sigma_r)$'s, denoised point clouds $\mathbf{Y}$.
```

cloud denoising

$$\min_{\mathbf{Y}} \sum_{i=1}^{3} [||\mathbf{X}_i - \mathbf{Y}_i||_2^2 + \alpha||\mathbf{X}_i - \mathbf{W}_i\boldsymbol{\sigma}||_2^2], \tag{5.31}$$

whose close-form solution for each coordinate is

$$\mathbf{Y}_i = [\mathbf{I} + \alpha(\mathbf{I} - \mathbf{P_s})^T(\mathbf{I} - \mathbf{P_s})]^{-1}\mathbf{X}_i. \tag{5.32}$$

Note that $\mathbf{P_s}$ is the supporting matrix. We then update the frequency components based on the denoised point clouds, and repeatedly carry out Step 1 to Step 3 until getting the final solution. In practice, we generally observe the convergence within only a few iterations. The complete algorithm is summarized in Algorithm 5 as shown in Fig. 5.5. Unlike for clear point clouds, we emphasize more on the smoothness of signals over the hypergraph. The parameter α can be set larger than used when dealing with clean point clouds.

5.4.4 Application Examples

In this section, we examine three application examples to test the efficacy of the proposed method in estimating hypergraph structure for both clear and noisy point clouds.

Sampling

Sampling is an important operation to facilitate analysis of very large point clouds. In this part, we consider different sampling strategies depending on different kinds of applications. Some interesting connections are found from the hypergraph frequency and point cloud features.

1) *Resampling using Harr-like Highpass Filtering*: Filtering helps extract select features of a given dataset. In some applications such as boundary detection, accurate extraction of shape features of point clouds is important. Thus, an efficient sampling should retain the features of the original point cloud. In our estimation of hypergraph structure, smoothness is a significant feature to model point clouds. Ideally, smoothness over the original surface of a point cloud should correspond to smoothness over its hypergraph model. Therefore, we can also design a Harr-like high-pass filter to extract sharp features over the surfaces.

Let $\mathbf{I}$ be an identity matrix of appropriate size. Similar to that in GSP [135], a Haar-like high-pass filter is designed as

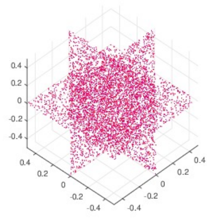
$$\mathbf{H} = \mathbf{I} - \mathbf{P_s} \tag{5.33}$$

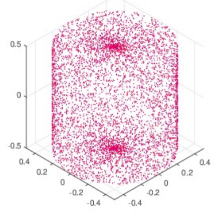
$$= \mathbf{V} \begin{bmatrix} 1-\sigma_1 & 0 & \cdots & 0 \\ 0 & 1-\sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1-\sigma_N \end{bmatrix} \mathbf{V}^T. \tag{5.34}$$

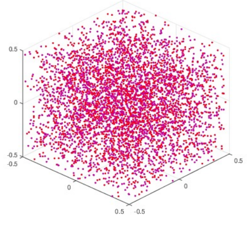
The filtered signal is

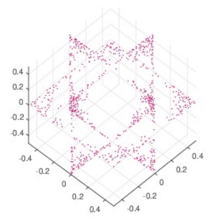
$$(\mathbf{Hs})_i = \mathbf{s}_i - \sum_j P_{s(ij)} \mathbf{s}_j, \tag{5.35}$$

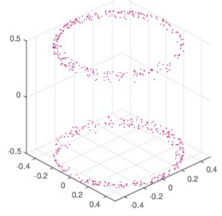
which reflects the differences between nodes and their neighbors over the hypergraph. Note that, the frequency coefficients together with their corresponding spectral components are ordered decreasingly here, i.e., $\sigma_i \geq \sigma_{i+1}$. From the definition of total variation, more smoothness corresponds to larger total variation. Thus, we can extract the sharp features

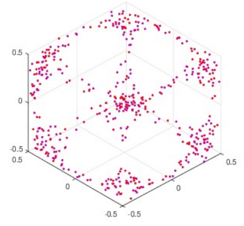

(a) Original Surfaces with 6000 Points.

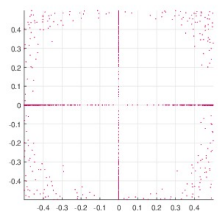

(b) Original Cylinder with 6000 Points.

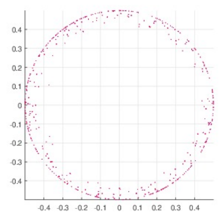

(c) Original Cube with 5000 Points.


(d) Sampled Surfaces with 800 Points.


(e) Sampled Cylinder with 600 Points.


(f) Sampled Cube with 500 Points.


(g) Sampled Surfaces with 800 Points.


(h) Sampled Cylinder with 600 Points.

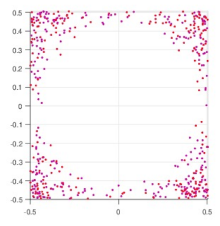

(i) Sampled Cube with 500 Points.

Figure 5.6: View from the Top of Sampled Point Clouds.

over the point clouds by sampling the nodes with large value of $||\mathbf{s}_i - \sum_j P_{s(ij)}\mathbf{s}_j||_2^2$.

To test this application, we estimate the spectral pairs for clean point clouds and filter the signals over several synthetic datasets. We randomly generate multiple points over the surfaces of basic graphics shown as Fig. 5.6(a) - 5.6(c), and sample the point clouds using the high-pass filter (HPF) given in Fig. 5.6(d) - 5.6(f). From the test results, we can see that the sampled points of the surfaces in Fig. 5.6(d) mainly congregate near the corners and edges, which are the sharp parts of the point clouds. In addition, the sampled nodes for a cube shape are also crowded near edges and corners. On the other hand, the sampled nodes of a cylinder are mostly at the boundaries of the cylinder. Our test results show that the Harr-like HPF can extract sharp features from point cloud surfaces, which correspond to the least smooth parts of the estimated hypergraph. Moreover, since the total variation measures the order of frequency, sharp features over the point cloud correspond to high frequency components. Thus, the hypergraph model and the estimated spectral pairs are efficient when extracting features of 3D point clouds.

2) *Down-Sampling with Hypergraph Fourier Transform*: Projecting signals into a suitable orthonormal basis is a widely-used sampling method [61]. The work of [123] develops a sampling theory based on hypergraph signal processing as follows:

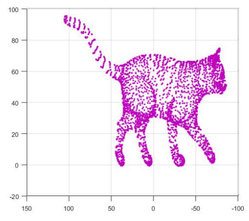
- Step 1: Order the spectrum components from low frequency to high frequency based on their total variations.
- Step 2: Implement hypergraph Fourier tranform as

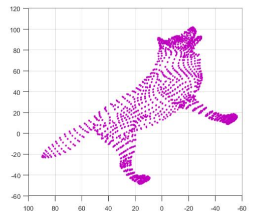
$$\mathcal{F}(\mathbf{s}) = [(\mathbf{f}_1^T\mathbf{s})^{M-1} \quad (\mathbf{f}_2^T\mathbf{s})^{M-1} \quad \cdots \quad (\mathbf{f}_N^T\mathbf{s})^{M-1}]^T. \tag{5.36}$$

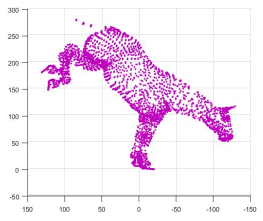
- Step 3: Use C transformed signal components in the hypergraph frequency domain to represent N signals in the original vertex domain.

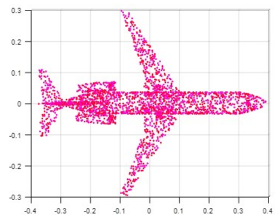
More specifically, for a K-bandlimitted hypergraph signal, a perfect recovery is available with K samples in hypergraph frequency domain. Similarly, we can sample the point clouds based on the hypergraph Fourier transform. To test the performance of the sampled signals, we implement hypergraph Fourier transform (HGFT) on each coordinates of the point clouds, i.e., $\mathcal{F}(\mathbf{X}_i)$ for all i. Then, we take the first C transformed signals in all coordinates. Finally, we implement the inverse hypergraph Fourier transform (iHGFT) to obtain the sampled shapes of the original point clouds. Note that, perfect recovery happens with C samples, if $(\mathcal{F}(\mathbf{X}_i))_{j+C} = 0$ for $i, j \in \mathcal{Z}^+$.

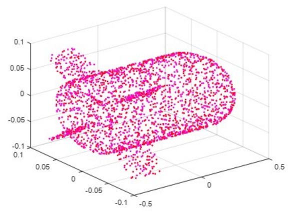
We test the recovered point clouds for animal point datasets [152–155] and the ShapeNet datasets of objects [156, 157] with the GSP-based methods. For the GSP-based method, we


(a) Cat.


(b) Wolf.


(c) Horse.


(d) Plane.


(e) Rocket.

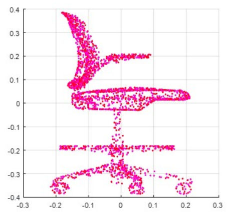

(f) Chair.

Figure 5.7: Test Datasets of Sampling.

construct a graph adjacency matrix $\mathbf{W}$ with Gaussian model, i.e.,

$$W_{ij} = \begin{cases} \exp\left(-\frac{||\mathbf{s}_i - \mathbf{s}_j||_2^2}{\delta^2}\right), & ||\mathbf{s}_i - \mathbf{s}_j||_2^2 \leq t; \\ 0, & \text{otherwise}, \end{cases} \tag{5.37}$$

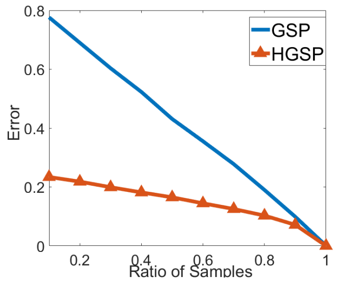
where $\mathbf{s}_i$ is the coordinates of the ith node. Then, we sample the point clouds using the signals after the graph Fourier transform (GFT).

The test point cloud is shown as Fig. 5.7. We first compare the error defined as

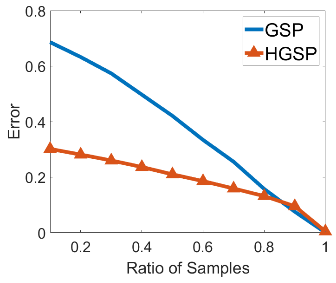
$$Error_1 = \frac{\sum_i |\mathbf{X}_i - \mathbf{X}_i^{rec}|}{\sum_i |\mathbf{X}_i|} \tag{5.38}$$

between the recovered point clouds and original point clouds with sampling ratio $0.1 \sim 1$ shown as Fig. 5.8.

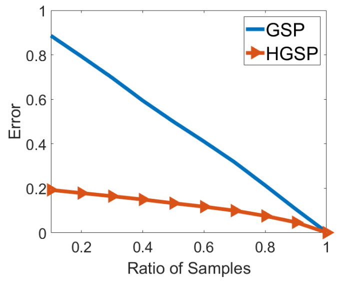
From the experimental results, we can see that the HGSP-based method has smaller error than the GSP downsampling method, clearly indicating hypergraph to be a better model. However, sometimes, the recovery error alone cannot tell the true story in terms of the performance for the recovered point clouds. To explore more, we compare the recovered



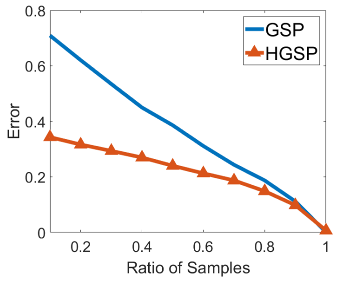
(a) Error for cat dataset.



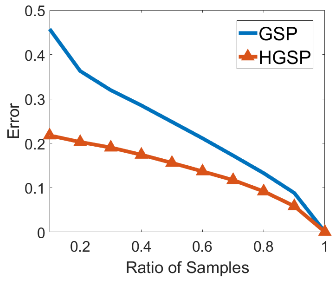
(b) Error for wolf dataset.



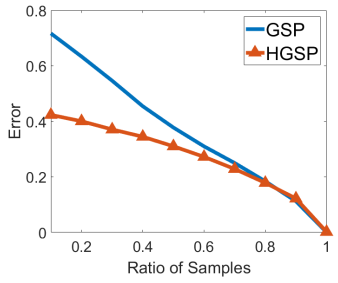
(c) Error for horse dataset.



(d) Error for plane dataset.



(e) Error for rocket dataset.



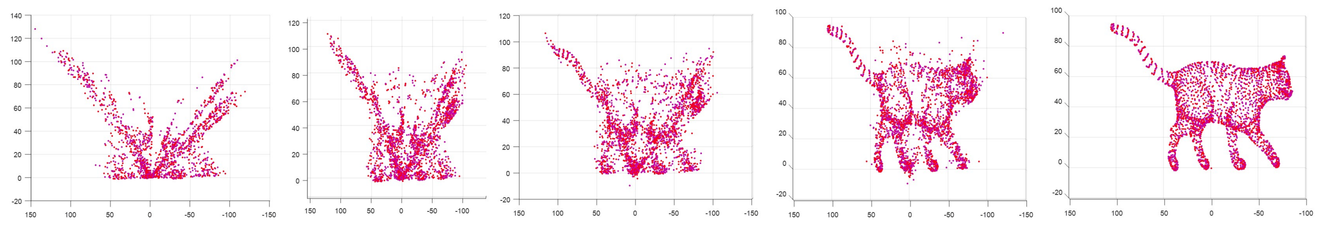
(f) Error for chair dataset.

Figure 5.8: Error between Recovered Data and Original Data.

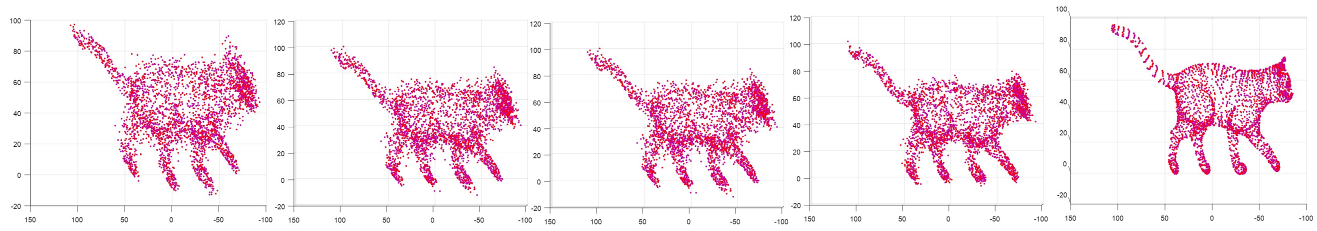
point clouds directly in Fig. 5.9. From the experimental results, we can see that HGSP-based method captures the overall structure of the point clouds with very few samples, whereas the GSP-based method requires more samples to get sufficient details. The error of GSP mainly stems from some outliers when taking more than 90 percent of the samples. The experiments show that HGSP-based method is a better tool for applications which need to recover an overall shape of point clouds from limited data storage. Our test shows hypergraph to be a suitable model for point clouds, and the estimated hypergraph spectral pairs capture the point cloud characteristics very well.

Denoising

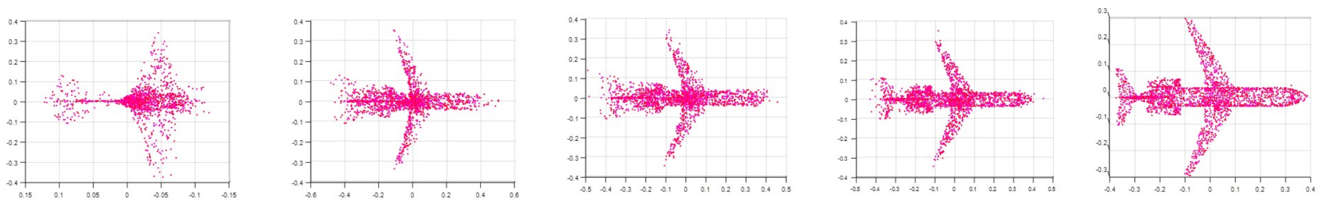
From estimated hypergraph spectral pairs from noisy point clouds, the performance of denoising is an intuitive metric of how good the estimates are. There are multiple methods developed to denoise noisy point clouds. The authors of [139] proposed a graph-based method to denoise based on total variation (GSP-TV). This method constructs a graph based on



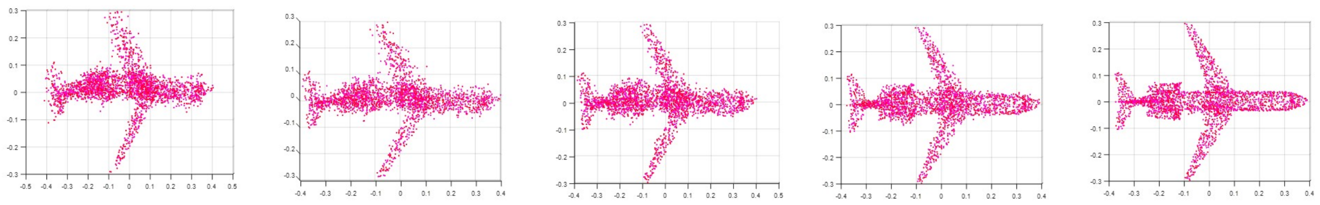
(a) Recovery from GSP-based Sampling with 30%, 50%, 70%, 90%, 98% Ratio of Samples for Cat Datasets.



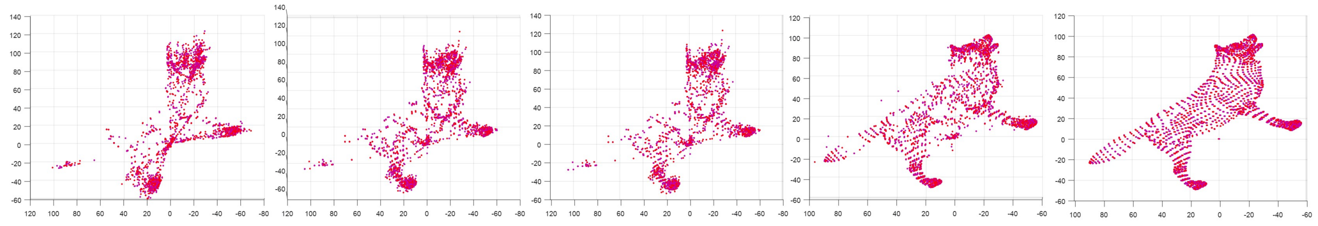
(b) Recovery from HGSP-based Sampling with 30%, 50%, 70%, 90%, 98% Ratio of Samples for Cat Datasets.



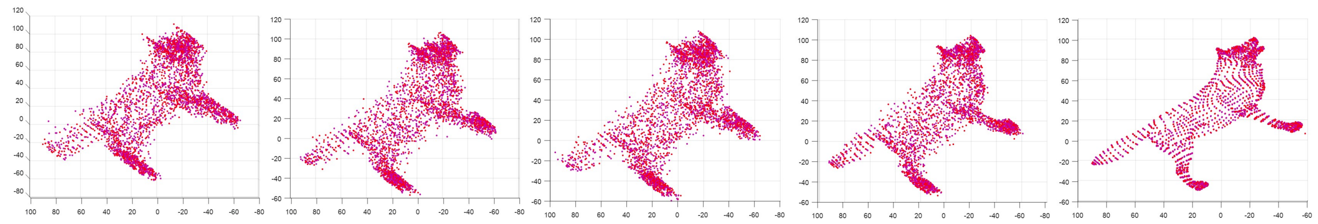
(c) Recovery from GSP-based Sampling with 30%, 50%, 70%, 90%, 98% Ratio of Samples for Plane Datasets.



(d) Recovery from HGSP-based Sampling with 30%, 50%, 70%, 90%, 98% Ratio of Samples for Plane Datasets.

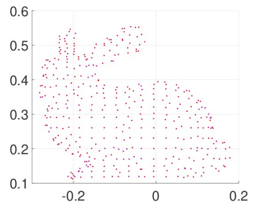


(e) Recovery from GSP-based Sampling with 30%, 50%, 70%, 90%, 98% Ratio of Samples for Wolf Datasets.

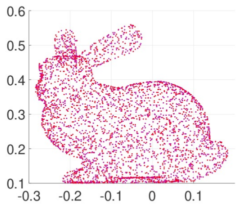


(f) Recovery from HGSP-based Sampling with 30%, 50%, 70%, 90%, 98% Ratio of Samples for Wolf Datasets.

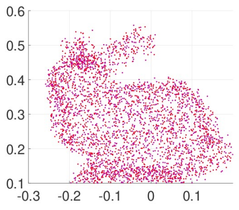
Figure 5.9: Recovered Point Clouds from Sampled Transformed Signals.



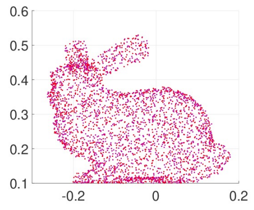
(a) Bunny with 397 Samples.



(b) Bunny with 3595 Samples.



(c) Noisy Bunny Dataset.



(d) Denoised Bunny Dataset.

Figure 5.10: Visualization of the Original and Denoised Bunny Point Clouds.

observed coordinates first before solving the denoising optimization

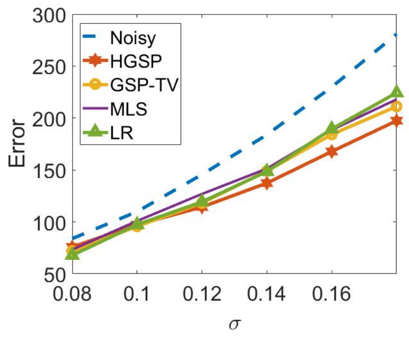
$$\min_{\mathbf{Y}} ||\mathbf{X}-\mathbf{Y}||_2^2 + \alpha \mathbf{TV}(\mathbf{Y}, \mathbf{W}), \tag{5.39}$$

where $\mathbf{X}$ is the observed coordinates, and $\mathbf{W}$ is the adjacency matrix. Here, the graph total variation $\mathbf{TV}(\mathbf{Y}, \mathbf{W})$ is applied in describing the smoothness over the graphs. In addition to total variation, Laplacian regularization (LR) [158] has also been used in denoising with a basic formulation

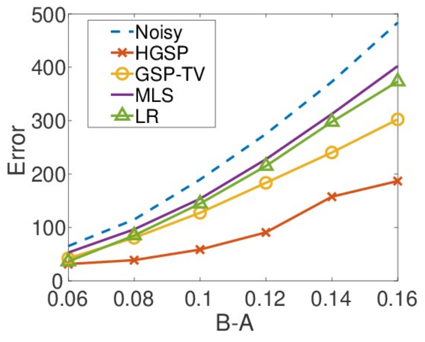
$$\min_{\mathbf{Y}} ||\mathbf{X}-\mathbf{Y}||_2^2 + \alpha ||\mathbf{Y}^T\mathbf{L}\mathbf{Y}||_2^2, \tag{5.40}$$

where $\mathbf{L}$ is the Laplacian matrix. Developed from traditional Laplacian regularization methods, a mesh Laplacian smooth (MLS) method is given in [159]. Other graph learning based methods include graph Laplacian regularization (GLR) [150] in a low-dimension manifold and feature graph learning (GL) [160].

1) *Overall Performance of Denoising*: To validate the performance of our denoising method, we first compare with the aforementioned traditional methods using the Standford bunny dataset with 397 points and sampled bunny with 3595 points, shown as Fig. 5.10(a) and Fig. 5.10(b), respectively. For GSP-based methods, a graph is constructed according to the Gaussian distance model to encode the local geometry information through an adjacency matrix in Eq. (5.37) [135]. For the Laplacian-based method, we establish the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{S}$, where $\mathbf{S}$ is the unweighted adjacency matrix and $\mathbf{D}$ is the diagonal



(a) Comparison in Gaussian Distribution.



(b) Comparison in Uniform Distribution.

Figure 5.11: Comparison between Different Methods.

matrix of node degree. We compare different methods in the sampled bunny dataset adding zero-mean Gaussian noise with variance σ^2, and zero-mean Uniform noise with the interval $B - A$, respectively. We use the error denoted by

$$Error_2 = \sum_{i=1}^{N} \sum_{j=1}^{3} |X_{ji} - Y_{ji}|, \tag{5.41}$$

where X_{ij} and Y_{ji} are the jth coordinates of observed and denoised point i, respectively, to measure the performance. We repeat the test on 1000 randomly generated noisy data. The error between the original dataset and the denoised dataset is shown in Fig. 5.11. The error of the noisy point clouds before denoising is also given as a reference in Fig. 5.11. From the test results, we can see that the HGSP-based method can achieve the lowest error, which demonstrates the effectiveness of the proposed denoising methods and estimated spectral pairs. In addition, the denoised bunny with 3595 samples is shown in Fig. 5.10(d), using our proposed method to denoise the noisy bunny in Fig. 5.10(c). The successful recovery of the bunny point cloud presents strong evidence that our estimated spectral pairs and denoising method are powerful tools in processing noisy point clouds.

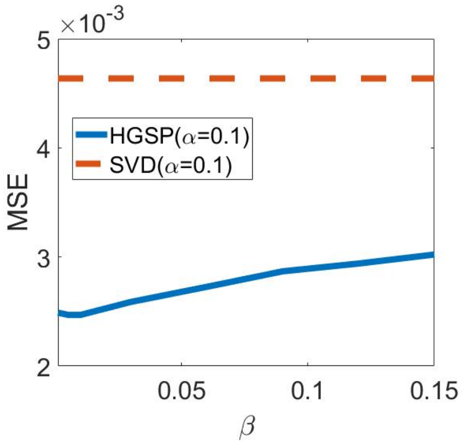
To test the performance in a more general experiment setup, we compare the proposed method with traditional methods together with graph learning based methods with other types of noise in both complex animal datasets and the ShapeNet object datasets in Fig. 5.7.

Table 5.2: Comparison in Different Datasets of Different Noise.

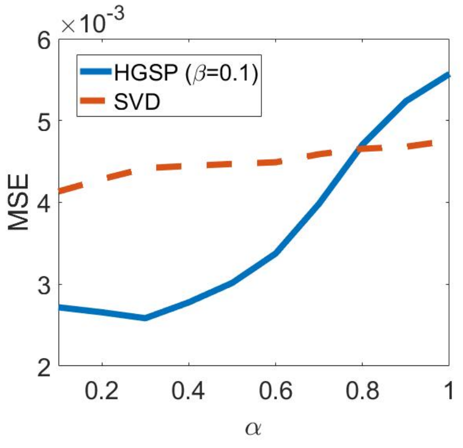
	Noisy	GSP-TV [8]	LR [39]	MLS [40]	GLR [41]	GL [42]	**HGSP**
Uniform∼U(-3, 3)							
Cat	2.9819	2.3744	2.4649	2.5312	2.2576	2.2412	**2.2209**
Horse	3.0377	2.7638	2.8280	2.8615	2.6777	2.6415	**2.6216**
Wolf	2.9857	2.6009	2.5650	2.6312	2.4391	2.4281	**2.4164**
Rocket	3.0587	2.6179	2.5391	2.6346	2.4512	2.3645	**2.3052**
Plane	2.9638	2.5369	2.4118	2.5110	2.4374	2.3827	**2.3690**
Chair	3.0233	2.5532	2.3718	2.5988	2.2855	**2.1809**	2.2048
Gaussian∼N(0, 2)							
Cat	4.0218	3.4952	3.2597	3.6202	**2.9778**	3.0112	3.0026
Horse	4.1020	3.6088	3.8241	3.7268	3.4370	3.3805	**3.2996**
Wolf	4.0402	3.4154	3.2637	3.4756	3.1616	3.0800	**3.0463**
Rocket	4.1527	3.2323	3.4771	3.5967	3.1415	**3.1063**	3.2471
Plane	3.9544	3.3188	2.9365	3.4101	2.9677	2.9246	**2.8868**
Chair	4.0119	3.4372	3.0770	3.5044	3.1987	3.0212	**3.0208**
Impulse (p=0.08)							
Cat	8.4066	**7.3728**	7.6285	7.8024	7.4001	7.3802	7.3844
Horse	35.7202	30.0853	29.8711	33.2011	28.9012	28.1285	**27.9143**
Wolf	9.4107	8.5086	8.5757	8.6105	8.2024	8.1684	**8.1395**
Rocket	1.8312	1.6718	1.6312	1.7354	1.6010	**1.5896**	1.6619
Plane	0.7759	0.6413	0.6559	0.6678	0.6752	0.6588	**0.6386**
Chair	1.1175	0.9374	0.9081	0.9661	0.8815	0.8622	**0.8598**
Average							
	5.5335	4.7318	4.6827	5.0047	4.5058	4.4195	**4.3657**

We normalize the coordinates of each point cloud to maintain a similar noise level. Using mean squared error (MSE) to measure the performances of different methods, the results are shown in Table. 5.2. We can see that the HGSP-based denoising achieves the best overall performances in most datasets with different noise types. Generally, GLR is better than traditional Laplacian regularization owing to its utilization of self-similarity among surface patches, and GL exhibits a slight improvement over GLR. The GSP-based total variation is worse than the HGSP-based total variation since the graph is constructed from noisy coordinates. Moreover, these methods perform better under Gaussian and uniform noises, since impulsive noise has a rescaling effect on the coordinates according to the coefficient p.

2) *Discussion*: Generally speaking, the total variation and the Laplacian regularization



(a) Impact of Sparsity.



(b) Impact of Smoothness.

Figure 5.12: Impact of Sparsity and Smoothness.

are both effective descriptions for signal smoothness. Hypergraphs and graphs can be efficient tools in processing point clouds for specific datasets. For example, hypergraphs are more efficient in processing complex shape features and tend to capture the overall shapes well with fewer samples. It would be interesting to explore a joint framework including both hypergraphs and graphs in processing different kinds of features in point clouds. Additionally, future works should also evaluate the effects of different tensor norms.

3) *Impact of Sparsity and Smoothness*: In addition to the overall performance, it is interesting to consider the ablation effect on smoothness and sparsity. To set up such additional experiments, we compare the impact of optimization of frequency coefficients with the simple approach of recovering eigenvalues from SVD of the covariance matrix.

We apply the same strategy to denoise the bunny datasets with uniform noise U(-0.1, 0.1) for the SVD-based method and the HGSP optimization based method in Eq. (5.29). For the SVD-based method, eigenvalues are directly from step 4 in Algorithm 3 and only α is used in denoising (smoothness) as Eq. (5.32). For the HGSP-based method, we first fix the parameter $\alpha = 0.1$ (smoothness) and measure the impact of β (sparsity) in Fig. 5.12(a). Next, we freeze the parameter $\beta = 0.1$ and test on different α's to measure the impact of smoothness in Fig. 5.12(b).

From Fig. 5.12, we observe that the optimal HGSP-based results achieve lower MSE than the SVD-based method in general, except for very large α. With properly chosen parameters for optimization, these results show that the HGSP-based method is better than the SVD-based method, and demonstrate the effectiveness of the proposed optimization-based method.

In Fig. 5.12(a), we can see that the appropriate β lies in $[0.01, 0.03]$, which is smaller than $\alpha = 0.1$. In Fig. 5.12(b), the optimum α lies near 0.3, larger than $\beta = 0.1$. Generally, α larger than β generates reasonably desirable results. Moreover, we can see from the curves of MSE that denoising is more sensitive to smoothness than sparsity.

Segmentation

One common processing task on 3D point clouds is unsupervised segmentation. The goal of point cloud segmentation is to identify points in a cloud with similar features for clustering into their respective regions [161]. These partitioned regions should be physically meaningful. Practical examples include the work of [162] which segments human posture point clouds for behavior analysis by partitioning human bodies into different semantic body parts. Segmentation facilitates point cloud analysis in various applications, such as object tracking, object classification, feature extraction, and feature detection.

Our proposed segmentation method targets gray-scale point clouds consisting of N points. There are three stages in the proposed segmentation: 1) estimate the hypergraph spectral space, 2) order and select the principal hypergraph spectrum, and 3) segment via clustering in the reduced hypergraph spectral space. In the first stage, instead of decomposing the constructed hypergraph, we estimate the hypergraph spectrum directly from the observed point clouds based on the hypergraph stationary process as Algorithm 3. This approach bypasses explicit hypergraph construction since the representing tensor is memory-inefficient

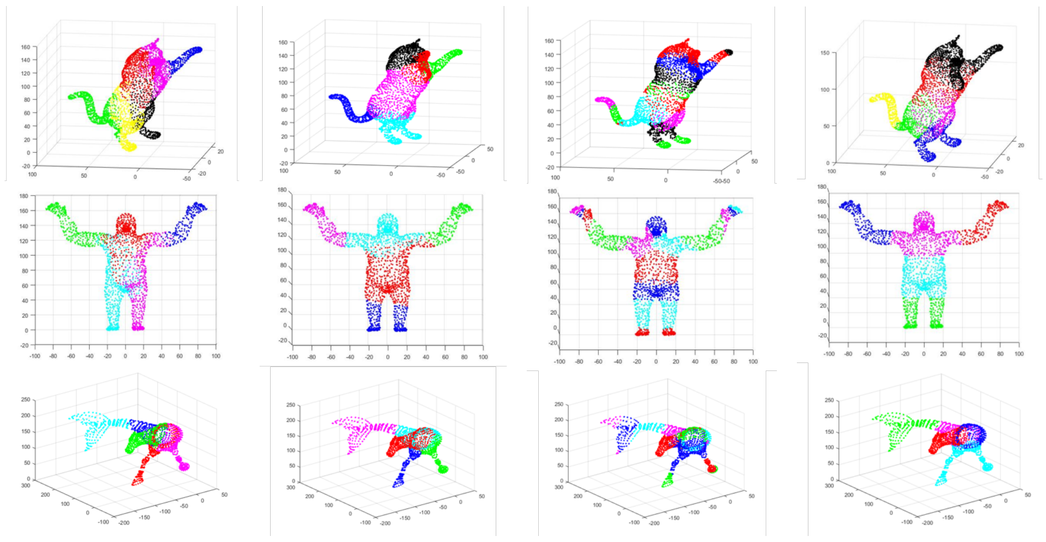
[5]Part of this chapter is reprinted, with permission, from [S. Zhang, S. Cui, and Z. Ding, "Point Cloud Segmentation based on Hypergraph Signal Processing," *IEEE Signal Processing Letters*, vol. 27, pp.1655-1659, Sep. 2020].

Algorithm 6 Hypergraph Spectral Clustering

```
1: Input: Point cloud dataset $\mathbf{s} = [\mathbf{X_1}, \mathbf{X_2}, \mathbf{X_3}] \in \mathbb{R}^{N\times 3}$ and the number of clusters $k$.
2: Calculate the mean of each row in $\mathbf{s}$, i.e., $\bar{\mathbf{s}} = (\mathbf{X_1} + \mathbf{X_2} + \mathbf{X_3})/3$;
3: Normalize the original point cloud data to zero-mean in each row, i.e., $\mathbf{s}' = [\mathbf{X_1} - \bar{\mathbf{s}}, \mathbf{X_2} - \bar{\mathbf{s}}, \mathbf{X_3} - \bar{\mathbf{s}}]$;
4: Calculate eigenvectors $\{\mathbf{f}_1, \cdots, \mathbf{f}_N\}$ for $R(\mathbf{s}') = \mathbf{s}'(\mathbf{s}'^H)$;
5: Estimate frequency coefficients $\sigma_r$'s by solving Eq. (5.26) approximately;
6: Rank frequency components $\mathbf{f}_r$'s based on their corresponding frequency coefficients $\sigma_r$ in the decreasing order.
7: Find the first $E$ leading spectral components $\mathbf{f}_r$ with larger $\sigma_r$ and construct a spectrum matrix $\mathbf{M} \in \mathbb{R}^{N\times E}$ with columns as the leading spectral components.
8: Cluster the rows of $\mathbf{M}$ using $k$-means clustering.
9: Cluster node $i$ into partition $j$ if the $i$th row of $\mathbf{M}$ is assigned to $j$th cluster.
10: Output: $k$ partitions of the point clouds.
```

and its orthogonal-CP decomposition is time-consuming. We then estimate the distribution of hypergraph frequency coefficients according to a measure of smoothness, and order the spectrum based on the hypergraph frequency. Finally, we identify the low frequency spectral contents and cluster in the optimized spectral space. The details of the algorithm is provided in Algorithm 6.

We test the performance of the proposed method along with traditional graph-based methods and k-means clustering. To implement k-means, we cluster over each row of the point cloud coordinates. For the hypergraph spectral clustering, we select the first E key spectral components according to frequency coefficients until a steep drop to the next (i.e., the $(E+1)$-th) coefficient. Typically, the first two or three elements sufficiently satisfy this criterion. When optimizing the frequency coefficients, an efficient β lies in 0.1-10 depending on the specific datasets. For the graph-based clustering, a Gaussian-graph model [135] is applied to encode the local geometry information through an adjacency matrix $\mathbf{W} \in \mathbb{R}^{N\times N}$. Let $\mathbf{s}_i \in \mathbb{R}^{1\times 3}$ be the ith point coordinate. The edge weight between points i and j is calculated as $W_{ij} = \exp\left(-\frac{||\mathbf{s}_i-\mathbf{s}_j||_2^2}{\delta^2}\right)$ if $||\mathbf{s}_i - \mathbf{s}_j||_2^2 \leq t$; otherwise, $W_{ij} = 0$. Here, the variance δ and the threshold t are parameters to control the edge weights. GSP spectrum is derived from the matrix $\mathbf{W}$. We also test the Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{S}$, where $\mathbf{S}$ is the unweighted adjacency matrix and $\mathbf{D}$ is the diagonal matrix of node degree.



(a) HGSP (b) GSP (c) Laplacian (d) Kmeans

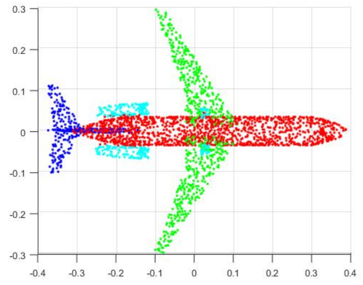
Figure 5.13: Results of Segmentation.

Table 5.3: Comparison in ShapeNet Datasets

	HGSP	GSP	Laplacian	K-means
Silhouette	**0.56748**	0.25756	0.137381	0.55894
Accuracy	**0.58928**	0.55321	0.502275	0.57699

1) *Overall Performance*: We first compare different methods in the animal datasets same as sampling. The overall results are shown in Fig. 5.13. The test results show that HGSP-based method, GSP-based method, and k-means clustering exhibit similar performance by clustering limbs and torsos. Interestingly, our HGSP method can further distinguish tails and different legs. Especially for the gorilla dataset in the second row of Fig. 5.13, HGSP spectral clustering segments different limbs with four different colors, whereas other methods fail to do so. We can see that the hypergraph model captures the overall structural information of 3D point clouds better than traditional graphs. The Laplacian-based method accentuates the details of some complex structures. For example, in the gorilla dataset, Laplacian-based method further distinguishes feet from legs and hands from arms, respectively. Generally, HGSP-based spectral clustering presents clearer segmentation of the main features for the point cloud datasets.

2) *Numerical Comparison*: To provide a comprehensive numerical comparison between


(a) Ground Truth.

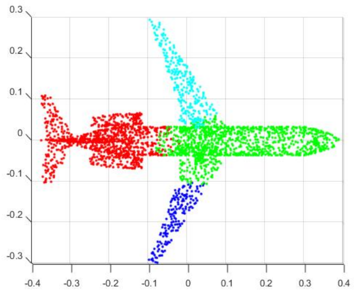

(b) HGSP Results.

Figure 5.14: Segmentation and Ground Truth.

different methods, we also compare the Silhouette index and mean accuracy of different methods in the ShapeNet Datasets. In the ShapeNet datasets, there are 16 categories of objects with labels in 2-6 classes. We test the average Silhouette and mean accuracy by randomly picking 50 point clouds from each category. The result is shown in Table 5.3. From the result, we can see that the HGSP-based method provides the largest Silhouette indices (indicating the best inner-cluster fitting) and the highest mean accuracy. Although these numerical results are valuable, a larger mean accuracy does not necessarily imply better performance in unsupervised clustering. For example, in Fig. 5.14, although the segmentation result differ from the ground truth, these results still make sense by grouping two wings to different classes. Often, visualization can be a more suitable performance assessment.

3) *Distribution of Eigenvalues*: We are interested in the reasons behind the performance differences of different graphical methods. To explore the reasons behind such differences, we examine the distributions of eigenvalues or the frequency coefficients of different methods in the specific horse point cloud shown in Fig. 5.15. In different rounds of the experiment, we randomly sample 400, 1400, 2400 and 3400 points from the original horse point cloud and calculate the eigenvalues from different methods. The results are shown in Fig. 5.16, Fig. 5.17 and Fig. 5.18. The Y-axis is the normalized eigenvalues or frequency coefficients. The X-axis is the eigenvalue order, i.e., $Pos_i = i/N$ for the ith eigenvalue of N nodes. From the results, we can see that the HGSP-based method and GSP-based method have quite similar

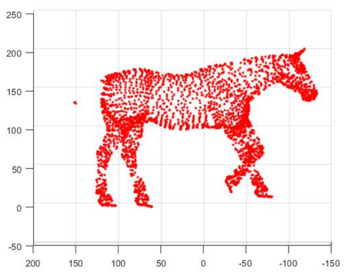
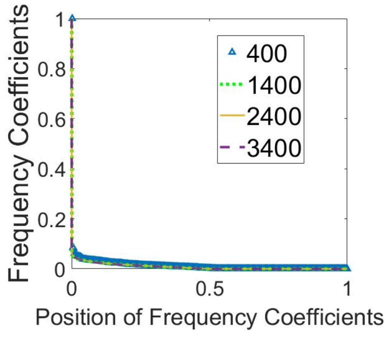
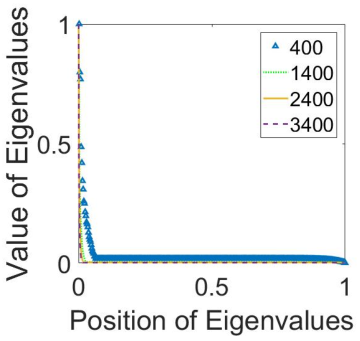


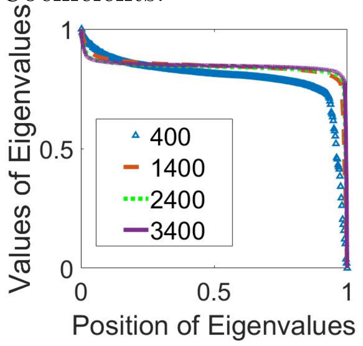
Figure 5.15: Horse Point Cloud.



m

Figure 5.16: HGSP Coefficients.





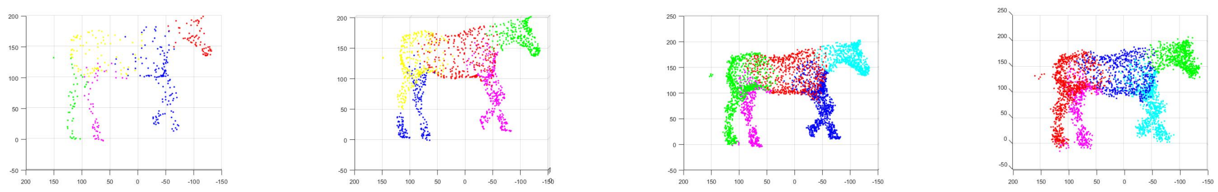
m

Figure 5.17: GSP Eigenvalues.

Figure 5.18: Laplacian Eigenvalues.

distributions, which indicate that their feature information is more concentrated in the first few key spectral components. Moreover, the HGSP-based method delivers a sharper curve than the GSP-based method. As mentioned in [107], a larger eigengap would lead to better clustering results, which should be responsible for the performance difference between the HGSP-based and GSP-based methods. Unlike adjacency-based methods, the distribution of eigenvalues of the Laplacian is rather different as shown in Fig. 5.18. In contrast to the Laplacian-based method, the HGSP-based method makes it easier to identify the vital information. This difference in eigenvalue distribution can account for the performance difference between the Laplacian-based segmentation and those based on adjacency.

4) *Complexity and Robustness*: We also test on datasets for different numbers of samples and noise effect. The results are shown in Fig. 5.19. The HGSP spectral clustering remains robust for either noisy data or down-sampled data. We compare the computation runtime of



(a) 400 Samples. (b) 1400 Samples. (c) SNR=32 dB. (d) SNR=25 dB.

Figure 5.19: Robustness of HGSP-based Segmentation.

Table 5.4: Running Time of Different Methods (in Seconds)

	Gorilla(2048 nodes)	Wolf(3400 nodes)	Cat(3400 nodes)
GSP	7.05	24.982	24.812
HGSP	2.771	11.451	11.335
Laplacian	4.662	15.579	15.773
k-Means	0.016	0.014	0.013

different methods over the animal datasets. From results summarized in Table 5.4, it is not surprising that the k-means method is the fastest, since graph-based methods require the additional step of spectrum estimation before clustering. The GSP-based and Laplacian-based methods require more computation, primarily because the computations needed to form the graph structure, whereas our proposed method directly estimates the HGSP spectral components. In particular, we only require an approximate distribution of the frequency coefficients to complete the segmentation task. Since the power of estimated coefficients is mainly concentrated in the first few spectral components shown as the optimized distribution in Fig. 5.16, a faster implementation can be done with the knowledge of the key estimated hypergraph spectra.

5.5 Conclusions

In this chapter, we introduce several fundamental definitions and properties of hypergraph frequency operations for image processing. We further present several application examples based on HGSP operations to illustrate the practical utility of HGSP in multimedia signal processing. Our goal is to develop new signal processing methods and analytical hypergraph tools for effective image processing and point cloud processing. Moreover, HGSP is also expected to be a useful tool in the development and analysis of HGCN.

Chapter 6

Graph Convolutional Networks based on Taylor Expansions

6.1 Introduction

Learning and signal processing over graph models have gained significant traction owing to their demonstrated abilities to capture underlying data interactions. Modeling each data point as a node and their interactions as edges in a graph, graph-based methods have been adopted in various signal processing and analysis tasks, such as semi-supervised classification [119, 163], spectral clustering [164, 165], link prediction [166] and graph classification [167]. For example, in [135], an undirected graph is constructed based on a Gaussian-distance model to capture geometric correlations among points in a point cloud, with which several graph-based filters have been developed to extract contour features of objects.

Among various graph-based tools, graph signal processing (GSP) has emerged as an efficient analytical tool for processing graph-modeled signals [2],[117]. Based on a graph Fourier space defined by the eigenspace of the representing adjacency or Laplacian matrix, GSP filters have found applications in practice, including bridge health monitoring [99], point cloud denoising [139], and image classification [73]. Leveraging graph Fourier transform [72], graph wavelet [116] and graph spectral convolution [127] can extract additional features from graph signals. For example, graph convolutional filters have been successful in edge detection and video segmentation [168].

Although GSP-based spectral filtering has demonstrated successes in a variety of applications, it still suffers from the high-complexity of spectrum computation and the need to select suitable propagation models. To efficiently extract signal features and integrate traditional GSP within the machine learning framework, graph convolutional networks (GCN) [119] have been developed for semi-supervised classification problems. Approximating graph spectral convolution with first-order Chebyshev expansions, GCN has been effective in such learning tasks. Furthermore, different GCN-related graph learning architectures, including personalized propagation of neural predictions (PPNP) [169] and N-GCN [170], have also been developed to process graph-represented datasets. However, some limitations remain with the traditional GCN based on Chebyshev expansions. For example, traditional GCN requires strong assumptions on maximum eigenvalues and Chebyshev coefficients for approximating spectral convolution, at the cost of possible information loss when compared against basic convolutional filters. Furthermore, systematic selection and design of graph representations for GCN remain elusive.

Our goal is to improve GCN by exploring its relationship with GSP. Specifically in this chapter, we explore the process from spectrum wavelet to vertex propagation, and investigate alternative designs for graph convolutional networks. Our contributions can be summarized as follows:

- We revisit graph spectral convolution in GSP and define conditions for approximating spectrum wavelet via propagation in the vertex domain. These conditions could provide insights to design GCN layers.
- We propose alternative propagation models for the GCN layers and develop a Taylor-based graph convolutional networks (TGCN) based on the aforementioned approximation conditions.

- We illustrate the effectiveness of the proposed frameworks over several well-known datasets in comparison with other GCN-type and graph-based methods in node signal classification.

- We also provide an interpretability discussion on the use of Taylor expansion, together with guidelines on selecting suitable graph representations for GCN layers.

In terms of the chapter organization, we first provide an overview on graph-based tools and review the fundamentals of GCN in Section 6.2 and Section 6.3, respectively. Next, we present the theoretical motivation and basic design of the Taylor-based graph convolutional networks (TGCN) in Section 6.4. Section 6.5 discusses the interpretability of Taylor expansion in approximation, and highlights the difference between TGCN and traditional GCN-based methods. We report the experimental results of the proposed TGCN framework on different datasets in Section 6.6, before concluding in Section 6.7.

6.2 Literature Review

In this section, we provide an overview on state-of-the-art graph signal processing (GSP) and graph convolutional networks (GCN).

6.2.1 Graph Signal Processing

Graph signal processing (GSP) has emerged as an exciting and promising new tool for processing large datasets with complex structures [2, 117]. Owing to its power to extract underlying relationships among signals, GSP has achieved significant success in generalizing traditional digital signal processing (DSP) and processing datasets with complex underlying features. Modeling data points and their interactions as graph nodes and graph edges, respectively, graph Fourier space could be defined according to the eigenspace of a graph representing matrix, such as the Laplacian or adjacency matrix, to facilitate data processing operations such as denoising [15], filter banks [16], and compression [17]. The framework of GSP can be further generalized over the graph Fourier space to include sampling theory [97], graph Fourier transform [72], frequency analysis [61], graph filters [73], graph wavelet [116] and graph stationary process [128]. In addition, GSP has also been considered for high-dimensional geometric signal processing, such as hypergraph signal processing [123] and topological signal processing [171].

6.2.2 Graph Convolutional Networks

Graph-based learning machines have become useful tools in data analysis. Leveraging graph wavelet processing [116], graph convolutional networks (GCN) approximate the spectral wavelet convolution via first-order Chebyshev expansions [119] and have demonstrated notable successes in semi-supervised learning tasks. Recent works, e.g., [169], have developed customized propagation of neural predictions (PPNP) to integrate PageRank [172] with GCNs. Other typical graph-based learning machines include GatedGCN [173], GraphSAGE [174], Gaussian Mixture Model Network (MoNet) [175], Graph Attention Networks (GAT) [176], Differential Pooling (DiffPool) [177], Geom-GCN [178], Mixhop [179], Diffusion-Convolutional Neural Networks (DCNN) [180], and Graph Isomorphism Network (GIN) [181]. For additional information, interested readers are referred to an extensive literature review [182] and a survey paper [12].

6.3 Graph Wavelet and Graph Convolutional Networks

In this section, we first review the fundamentals of graph spectral convolution and wavelets, necessary for the development of propagation models of the GCN layers. We will then briefly introduce the structures of traditional GCN [119]. For convenience, some of the important notations and definitions are illustrated in Table 6.1.

Table 6.1: Notations and Definitions

Notation	Definition
$\mathcal{G} = (\mathcal{V}, \mathcal{E})$	The undirected graph
$\mathcal{V}$	The set of nodes in the graph $\mathcal{G}$
$\mathcal{E}$	The set of edges in the graph $\mathcal{G}$
$\mathbf{A}$	The adjacency matrix
$\mathbf{L}$	The Lapalcian matrix
$\mathbf{P}$	The general propagation matrix of the graph $\mathcal{G}$
$\mathbf{D}$	The diagonal matrix of node degree
λ	The eigenvalue of the propagation matrix
λ_{max}	The maximal eigenvalue
$\mathbf{V}$	The spectral matrix with eigenvectors of $\mathbf{P}$ as each column
$\mathbf{x}$	The graph signal vector
$\mathbf{X}^{(l)}$	The feature data at l layer
$\mathbf{I}$	The identity matrix

6.3.1 Graph Spectral Convolution and Wavelet-Kernels

An undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $N = |\mathcal{V}|$ nodes can be represented by a representing matrix (adjacency/Laplacian) decomposed as $\mathbf{P} = \mathbf{V}\boldsymbol{\Sigma}\mathbf{V}^T \in \mathbb{R}^{N\times N}$, where the eigenvectors $\mathbf{V} = \{\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_N\}$ form the graph Fourier basis and the eigenvalues λ_i's represent graph frequency [72].

In GSP [126,127], graph Fourier transform of convolution between two signals is a product between their respective Fourier transforms denoted by $\diamond$, i.e.,

$$\mathbf{x} \diamond \mathbf{y} = \mathcal{F}_C^{-1}(\mathcal{F}_C(\mathbf{x}) \circ \mathcal{F}_C(\mathbf{y})), \tag{6.1}$$

where $\mathcal{F}_C(\mathbf{x}) = \mathbf{V}^T\mathbf{x}$ refers to the graph Fourier transform (GFT) of signals $\mathbf{x}$, $\mathcal{F}_C^{-1}(\hat{\mathbf{x}}) = \mathbf{V}\hat{\mathbf{x}}$ is the inverse GFT and $\circ$ is the Hadamard product. This definition generalizes the property that convolution in the vertex domain is equivalent to product in the corresponding graph spectral domain.

In [116], the graph wavelet transform is defined according to graph spectral convolution. Given a spectral graph wavelet-kernel $\hat{\mathbf{g}} = [g(\lambda_1), g(\lambda_2), \cdots, g(\lambda_N)]^T$ with kernel function

$g(\cdot)$, the graph wavelet operator is defined as

$$T_g(\mathbf{x}) = \mathbf{V}(\hat{\mathbf{g}} \circ (\mathbf{V}^T\mathbf{x})) \tag{6.2}$$

$$= \mathbf{V}\begin{bmatrix} g(\lambda_1) & \cdots & 0 \\ 0 & \ddots & 0 \\ 0 & \cdots & g(\lambda_N) \end{bmatrix}\mathbf{V}^T\mathbf{x}. \tag{6.3}$$

Note that graph wavelet can be interpreted as a graph convolutional filter with a spectrum wavelet-kernel $\hat{\mathbf{g}}$. Depending on the datasets and applications, different kernel functions may be utilized in Eq. (6.3).

6.3.2 Graph Convolutional Networks and Their Limitations

To overcome the complexity for computing the spectrum matrix $\mathbf{V}$ and the difficulty of seeking suitable wavelet-kernel functions, one framework of GCN developed in [119] considers a first-order Chebyshev expansion. Considering Chebyshev polynomials $T_K(x)$ up to K^{th} orders and the Laplacian matrix as the propagation matrix, the graph convolutional filter with wavelet-kernel $\hat{\mathbf{g}}$ is approximated by

$$T_g(\mathbf{x}) \approx \sum_k \theta_k T_k(\tilde{\mathbf{L}})\mathbf{x}, \tag{6.4}$$

where $\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{\max} - \mathbf{I}_N$. With careful choice of $\lambda_{\max}$ and parameters θ_k, the graph convolutional filter can be further approximated by the 1st-order Chebyshev expansion

$$T_g(\mathbf{x}) \approx \theta(\mathbf{I}_N + \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}})\mathbf{x}, \tag{6.5}$$

where $\mathbf{D}$ is the diagnal matrix of node degree. From here, by generalizing the approximated graph convolutional filter to a signal $\mathbf{X} \in \mathbb{R}^{N\times C}$ with C features for each node, the filtered signals can be written as

$$\mathbf{Z} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{\Theta}, \tag{6.6}$$

where $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$, $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$, and $\mathbf{\Theta} \in \mathbb{R}^{C \times F}$ is the parameter matrix. Furthermore, by integrating the nonlinear functions within the approximated convolutional filters, a two-layer GCN can be designed with message propagation as

$$\mathbf{Z}_{GCN} = \text{softmax}\left(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\ \text{RELU}(\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}\mathbf{W}^{(0)}))\mathbf{W}^{(1)}\right),$$

where $\mathbf{W}^{(0)} \in \mathbb{R}^{N \times H}$ and $\mathbf{W}^{(1)} \in \mathbb{R}^{H \times C}$ are the parameters for the H hidden units. Here we use standard terminologies of "softmax" and "RELU" from deep learning neural networks.

Although GCN has achieved success in some applications, some drawbacks remain. First, it relies on several strong assumptions to approximate the original convolutional filters. For example, $\lambda_{\max} = 2$ are used to approximate in implementation due to the range of variables in Chebyshev expansions, and the Chebyshev coefficients are set to $\theta_1 = -\theta_0 = -\theta$ to obtain Eq. (6.6). These assumptions may compromise the efficacy of spectral convolution. Second, the graph representation $\tilde{\mathbf{D}}^{\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{\frac{1}{2}}$ may not always be the optimal choice while the Chebyshev approximation limits the type of representing matrix. We provide a more detailed discussion in Section 6.5. In addition, it remains unclear as to how best to derive a suitable kernel-function $\hat{\mathbf{g}}$ and its approximation. Moreover, insights in terms of interpretability is highly desirable from spectral wavelet convolution to vertex propagation.

To explore alternatives in designing propagation model for GCNs, we focus on the process between graph spectral wavelet-kernels and propagation in the vertex domain. We will further propose alternative propagation models for GCNs.

6.4 Taylor-based Graph Convolutional Networks

In this section, we investigate conditions needed for approximating the spectral convolution via vertex propagation. Next, we propose alternative propagation models for graph convolution layers based on Taylor expansion, where the general convolutional filter can be written as

$$\mathbf{Z} = G_\alpha(\mathbf{P})\mathbf{X}\boldsymbol{\Theta}, \tag{6.7}$$

where $G_\alpha(\mathbf{P})$ is a polynomial function with parameter α, $\mathbf{P}$ is the representing matrix of the graph, and $\boldsymbol{\Theta}$ are parameters of feature projection.

6.4.1 Approximation of Spectral Convolution

We first present the theoretical motivation for designing a polynomial-based propagation model, and its relationship to the graph spectral wavelets. For a polynomial filter in GSP, let $\mathbf{P}$ be the representing (adjacency/Laplacian) matrix. We can obtain the following property.

Lemma 6.1. *Given a GSP polynomial filter $\mathbf{H} = h(\mathbf{P}) = \sum_k \alpha_k \mathbf{P}^k$, the filtered signals are calculated by*

$$\mathbf{H}\mathbf{x} = h(\mathbf{P})\mathbf{x} = \sum_{r=1}^{N} h(\lambda_r)\mathbf{f}_r(\mathbf{f}_r^{\mathrm{T}}\mathbf{x}), \tag{6.8}$$

where $\mathbf{f}_r$'s are the graph spectrum and λ_r's are the eigenvalues of $\mathbf{P}$ related to graph frequency.

This lemma shows that the response of the filter to an exponential is the same exponential amplified by a gain that is the frequency response of the filter at the frequency of the exponential [2]. The exponentials are the eigenfunctions/eigenvectors, similar to complex exponential signals in linear systems.

Looking into the graph wavelet convolutional filter in Eq. (6.3), the wavelet-kernel function $g(\cdot)$ operates to modify frequency coefficients λ_r's. Thus, we have the following property of transferring spectrum wavelet to vertex propagation.

Theorem 6.1. *Given a polynomial wavelet kernel function $g(\cdot)$, the GSP convolutional filter on signal* $\mathbf{x}$ *is calculated as*

$$T_g(\mathbf{x}) = g(\mathbf{P})\mathbf{x}. \tag{6.9}$$

This theorem indicates that we can bypass computing the spectrum by implementing the convolution directly in vertex domain, since the wavelet kernel $g(\cdot)$ is polynomial or can be approximated by a polynomial expansion. We can see that the Chebyshev expansion is a special case of *Theorem 6.1.* In addition to Chebyshev expansion, Legendre [183] and Taylor [184] expansions can also approximate the spectral convolution. In addition, other polynomial design on the wavelet-function $g(\cdot)$ are also possible.

6.4.2 Taylor-based Propagation Model

We now provide alternative propagation models for the GCN layers based on Taylor expansions, with which the wavelet-kernel function $g(x)$ can be approximated via

$$g(x) \approx \sum_{k=0}^{K} \theta_k (x-a)^k. \tag{6.10}$$

Here, $\theta_k = g^{(k)}(a)/n!$ is the expansion coefficients.

Since the Taylor approximation of $g(x)$ in Eq. (6.10) is a polynomial function of the variable x which meets the condition in *Theorem 6.1*, the graph spectral convolutional filter can be approximated as

$$T_g(\mathbf{x}) \approx \sum_{k=0}^{K} \theta_k (\mathbf{P} - \operatorname{diag}(\boldsymbol{\Phi}))^k \mathbf{x}, \tag{6.11}$$

where $\boldsymbol{\Phi}$ is a generalization of the parameter a. We will discuss further the intuition of applying Taylor expansion and its difference with Chebyshev approximation in Section 6.5.

We can develop different models based on Eq. (6.11) to develop the Taylor-based GCN (TGCN). The $\mathbf{P}$ matrix here can be any practical graph representing matrix used to capture overall information of the graph. For example, typical representing matrices include the adjacency matrix $\mathbf{A}$, the Laplacian matrix $\mathbf{L}$, or the normalized propagation matrix $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$. Section 6.6 provides further discussions on the selection of representing matrix.

We now provide several types of TGCN design.

Type-1 First-Order TGCN: Similar to the traditional GCN, we first consider TGCN based on the first-order Taylor expansions with a simpler diagonal matrix $\text{diag}(\mathbf{\Phi}) = \phi\mathbf{I}_N$. Letting $K = 1$, Eq. (6.11) can be written as

$$T_g(\mathbf{x}) \approx [(\theta_0 - \theta_1\phi)\mathbf{I}_N + \theta_1\mathbf{P}]\mathbf{x} \tag{6.12}$$

$$= \theta'(\mathbf{P} + \alpha\mathbf{I}_N)\mathbf{x}, \tag{6.13}$$

where $\theta' = \theta_1$ and $\alpha = \frac{\theta_0 - \theta_1\phi}{\theta_1}$ are the new parameters for the convolutional filter. As a result, the GCN layer with generalized signal $\mathbf{X} \in \mathbb{R}^{N\times C}$ can be designed as

$$\mathbf{X}^{(l+1)} = \sigma\{(\mathbf{P} + \alpha_l\mathbf{I}_N)\mathbf{X}^{(l)}\mathbf{\Theta}_l\} \tag{6.14}$$

where α_l and $\mathbf{\Theta}_l$ are the trainable variables for the l^{th} layer, and $\sigma\{\cdot\}$ is the activation function.

Type-2 First-Order TGCN: We also consider more general diagonal matrix in place of $\alpha\mathbf{I}_N$, i.e.,

$$\mathbf{X}^{(l+1)} = \sigma\{(\mathbf{P} + \text{diag}(\boldsymbol{\beta}_l))\mathbf{X}^{(l)}\mathbf{\Theta}_l\}, \tag{6.15}$$

where $\boldsymbol{\beta}_l$ and $\mathbf{\Theta}_l$ are the parameters of the l^{th} layer. Here, the self-influence for each node varies from node to node, whereas each node affects itself equivalently in the type-1 first-order TGCN model.

Type-3 k^{th}-Order TGCN: We also consider the higher-order polynomial propagation models for each layer. To avoid overfitting and reduce the complexity, we require $\theta_k = \theta$ for all k and the diagonal matrix as $\alpha \mathbf{I}_N$ in Eq. (6.11). Then the resulting TGCN layer becomes

$$\mathbf{X}^{(l+1)} = \sigma\{[\sum_k (\mathbf{P} + \alpha_l \mathbf{I}_N)^k]\mathbf{X}^{(l)}\mathbf{\Theta}_l\}. \tag{6.16}$$

Compared to the 1st-order approximation, the higher-order approximation contains more trainable parameters and computations, resulting in higher implementation complexity.

Type-4 k^{th}-Order TGCN: More general TGCN layers can be designed without requiring $\theta_k = \theta$ for all k as follows:

$$\mathbf{X}^{(l+1)} = \sigma\{\sum_k [(\mathbf{P} + \alpha_l \mathbf{I}_N)^k \mathbf{X}^{(l)}\mathbf{\Theta}_{l,k}]\}. \tag{6.17}$$

Here, we only consider simple diagonal with one parameter α in the higher-order polynomials to avoid overfitting and high complexity. We will provide some insights into the choice of different approximation models in Section 6.6.

6.5 Discussion

In this section, we discuss the interpretability of the Taylor approximation of wavelet-kernels, and illustrate its differences with the existing GCNs.

6.5.1 Interpretation of the Graph Convolution Approximation

To understand the connection between the graph convolution filters and the approximated GCNs, we start from the basic graph wavelet operator in Eq. (6.3). Given the definition of graph convolution in Eq. (6.1) and the graph spectrum matrix $\mathbf{V} = [\mathbf{f}_1, \mathbf{f}_2, \cdots, \mathbf{f}_N]$, the graph wavelet operator is a convolution-based filter on the signal $\mathbf{x}$ with a parameter vector $\boldsymbol{\delta}$, and is denoted by

$$T_g(x) = \boldsymbol{\delta} \diamond \mathbf{x}, \tag{6.18}$$

where the graph Fourier transform of $\boldsymbol{\delta}$ requires $\mathbf{V}^T\boldsymbol{\delta} = \hat{\mathbf{g}}$. Suppose that $\mathbf{f}_i$ is the corresponding eigenvector of λ_i. Each term in the wavelet-kernel (i.e., $g(\lambda_i) = \mathbf{f}_i^T\boldsymbol{\delta}$) embeds the information of graph spectrum with the parameters $\boldsymbol{\delta}$ by the function $g(\cdot)$. Thus, the optimization on the wavelet-kernel $g(\cdot)$ is equivalent to finding suitable parameters for the convolution filter to achieve the goals, such as minimizing the error between the filtered signal and its labels.

Computing the exact graph spectra can be time-consuming. Nevertheless, GCNs and TGCNs approximate the graph-kernel by polynomial expansions, i.e.,

$$T_g(\mathbf{x}) \approx \sum_k \theta_k T_k(\mathbf{P})x, \tag{6.19}$$

where θ_k is the expansion coefficients and $T_k(\cdot)$ is the k-th term of polynomials. Here, the θ_k is a function of $g(\cdot)$, i. e.,

$$\theta_k = g^{(k)}(a)/n! \tag{6.20}$$

for Taylor polynomials, and

$$\theta_k = \frac{2}{\pi}\int_{-1}^{-1} \cos(k\theta)g(\cos(\theta))d\theta \tag{6.21}$$

for Chebyshev polynomials [116].

Since $T_k(\mathbf{P})$ is already determined for each type of expansions, the optimization on the wavelet-kernel function $g(\cdot)$ is transformed into estimating the polynomial coefficients θ_k, i.e., $\boldsymbol{\Theta}$, in the GCN propagation layers. More specifically, although we may need prior knowledge on the derivatives of $g(\cdot)$ for Taylor coefficients, $g^{(k)}(a)$ can be reparametrized as the parameters θ_k of the convolution filter, given our goal to optimize the function $g(\cdot)$ by using deep learning networks. The information of the Taylor expansions remains in the polynomials terms, i.e.,

$$T_k(\mathbf{P}) = (\mathbf{P} - \mathrm{diag}(\boldsymbol{\Phi}))^k, \tag{6.22}$$

which differs from the Chebyshev expansions.

6.5.2 Comparison with Chebyshev-based GCNs

Now, we compare the differences between Chebyshev-based GCNs and Taylor-based GCNs to illustrate the benefits of Taylor expansions.

For the Chebyshev expansion, variable x is bounded within $[-1, 1]$. In the graph wavelet-kernel, each λ is within $[0,\ \lambda_{\text{max}}]$ for the Laplacian matrix. A simple transformation $\lambda = a(y + 1)$, with $a = \lambda_{max}/2$ [116] can change variables from λ to y:

$$y = \frac{2\lambda}{\lambda_{max}} - 1, \tag{6.23}$$

which accounts for the use of graph representation, i. e.,

$$\tilde{\mathbf{L}} = 2\mathbf{L}/\lambda_{\text{max}} - \mathbf{I}_N, \tag{6.24}$$

in Eq. (6.4) for Chebyshev-based GCNs. Unlike Chebyshev expansion, Taylor polynomials do not limit variable x to an interval (without using λ_{max}). For this reason, TGCN admits a more flexible design of graph representation $\mathbf{P}$ without limiting the intervals of eigenvalues. There is no need to set the value of the largest eigenvalue or normalize the graph representation when implementing TGCNs. We shall test different graph representations when we present the experiment results next.

In addition, the Taylor polynomial gives a more unified simple design for polynomial terms $T_k(\mathbf{P})$. In Chebyshev polynomials, each polynomial term is recursive from its previous terms, thereby making it less expedient to implement higher-order GCN. However, Taylor polynomials take the same form regardless of a and are regular. The additional parameters $\mathbf{\Phi}$ also provide benefits for TGCN. In signal propagation, parameters $\mathbf{\Phi}$ can be interpreted as the self-influence from the looping effects. In practical applications, such self-influence does occur and may be less obvious within the data. For example, in the citation networks, the work from highly-cited authors may have greater impact and trigger the appearance of a series of related new works on its own, which indicates larger self-influence as well as higher impact on other works. We will illustrate this impact further in Section 6.6.

6.5.3 Single-layer High-order vs. Multi-layer First-order

From the design of the first-order TGCN, multiple layers of the first-order propagation also forms a higher-order polynomial design. However, such design with multiple layers of the first-order polynomials is different from the single-layer high-order propagation. Suppose that a single-layer k^{th}-order polynomial convolutional filter is written as

$$\mathbf{Z}_k = \sum_k \alpha_k (\mathbf{P} + \text{diag}(\boldsymbol{\beta}))^k \mathbf{X}\boldsymbol{\Theta} \tag{6.25}$$

and the first-order polynomial is

$$\mathbf{Z}_1 = \alpha(\mathbf{P} + \text{diag}(\boldsymbol{\beta}))\mathbf{X}\boldsymbol{\Theta}. \tag{6.26}$$

For a k-layer first-order polynomial convolutional filter, the filtered result can be written as

$$\begin{aligned} \mathbf{Z}^{(k)} &= \alpha_1 \cdots \alpha_k (\mathbf{P} + \text{diag}(\boldsymbol{\beta}))^k \mathbf{X}\boldsymbol{\Theta}_1 \cdots \boldsymbol{\Theta}_k & (6.27) \\ &= \alpha' (\mathbf{P} + \text{diag}(\boldsymbol{\beta}))^k \mathbf{X}\boldsymbol{\Theta}', & (6.28) \end{aligned}$$

which is one term in the single-layer k^{th}-order polynomial convolutional filter. Thus, the multi-layer first-order TGCN is a special case of single-layer higher-order polynomials. Since the high-order polynomial designs have already embedded the high-dimensional propagation of signals over the graphs, we usually apply the single-layer design instead of multi-layer for higher-order TGCNs in implementation to reduce complexity.

Table 6.2: Data Statistics

Datasets	Number of Nodes	Number of Edges	Number of Features	Number of Classes	Label Ratio
Cora	2708	5728	1433	7	0.052
Citeseer	3327	4614	3703	6	0.036
Pumbed	19717	44325	500	3	0.0031

6.6 Experiments

We now test the TGCN models in different classification experiments. We first measure the influence of depth and polynomial orders in different types of TGCNs in citation networks. Next, we experiment with different representing matrices and propagation models to explore the choice of suitable layer design and graph representations. We also report comparative results with other GCN-like methods in node classification and demonstrate the practical competitiveness of our newly proposed framework.

6.6.1 Evaluation of Different TGCN Designs

In this subsection, we evaluate different designs of TGCN to show its overall performance in node classification of citation networks. Additional comparisons with other existing methods are provided in Section 6.6.2.

Experiment Setup

We first provide the TGCN experiment setup.

Datasets: We use three citation network datasets for validation, i.e., Cora-ML [185,186], Citeseer [187], and Pubmed [188]. In these citation networks, published articles are denoted as nodes and their citation relationships are represented by edges. The data statistics of these citation networks are summarized in Table 6.2. We randomly select the a subset from the original datasets with 10% training data, 60% test data and 30% validation data.

Convolution Layer: For the first-order TGCN, we consider a two-layer structure de-

Table 6.3: Overall Accuracy for Different First-Order Methods (Percent)

Methods	Representing Matrix	Cora	Citeseer	Pubmed
GCN	$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$	78.7±2.7	68.7±2.7	78.8±3.1
Type-1 First-Order TGCN[1]	$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$	79.9±1.7	70.1±1.5	79.3±2.0
Type-1 First-Order TGCN[2]	$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$	78.8±3.4	68.9±1.4	78.9±2.5
Type-2 First-Order TGCN	$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$	$\mathbf{81.3 \pm 2.7}$	$\mathbf{70.3 \pm 2.6}$	$\mathbf{79.8 \pm 2.6}$

* For type-1 first-order TGCN with propagation model $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, we adjust parameters both manually and automatically. The results are reported in [1] for manual and [2] for automatic adjustments, respectively.

signed as follows.

$$\mathbf{Z} = \text{softmax}(G_{\Phi_1}(\mathbf{P})\text{RELU}(G_{\Phi_0}(\mathbf{P})\mathbf{X}\mathbf{W}^{(0)})\mathbf{W}^{(1)}), \tag{6.29}$$

where $G_{\Phi_1}(\mathbf{P})$ is the specific type of TGCN propgation model, and $\mathbf{W}^{(0)} \in \mathbb{R}^{N\times H}$ together with $\mathbf{W}^{(1)} \in \mathbb{R}^{H\times C}$ are the parameters of the H hidden units. For the higher-order TGCN, we consider a single-layer structure, i.e.,

$$\mathbf{Z} = \text{softmax}(G_{\Phi}(\mathbf{P})\mathbf{X}\mathbf{W}). \tag{6.30}$$

When training the parameters, we let the neural networks learn the diagonal parameters $\boldsymbol{\beta}$ for type-2 first-order TGCN, and α for higher-order TGCN. We applied Adam optimizer [189] for network training. For type-1 first-order TGCN, we apply both manual and automatic adjustments on the diagonal parameters α. We train the projection parameter $\mathbf{W}$ for a variety of TGCNs. For graph representing matrices, we first apply the normalized $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$ to measure the effects of layer depth, polynomial orders, and propagation models, respectively. We then test the results of different representing matrices $\mathbf{P}$ to gain insights and guidelines on how to select suitable graph representations.

Implementation: Let $\mathcal{V}_l$ be the set of labeled examples and Y_i denote the labels. We

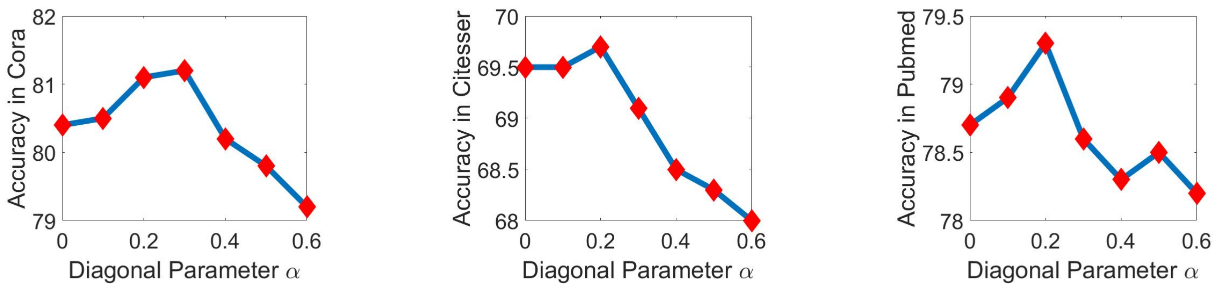


Figure 6.1: Optimal α for Type-1 First-Order TGCN.

evaluate the cross-entropy error over all labeled examples to train parameters, i.e.,

$$\mathcal{L} = -\sum_{i \in \mathcal{V}_l} \sum_{j=1}^{L} Y_{ij} \ln Z_{ij}. \tag{6.31}$$

Hyperparameter: For fair comparison of different designs of TGCN, we use the similar hyperparameters, with dropout rate $d = 0.5$, learning rate $r = 0.01$ and weight decay $w = 5 \times 10^{-4}$. For two-layer TGCNs, we let the number of hidden units be $H = 40$. For the higher-order TGCNs, we use fewer hidden units to reduce the complexity.

Performances of Different TGCN Propagation Models

We first measure the performances of first-order TGCNs by comparing different propagation models against the traditional GCNs. Note that, the type-1 first-order TGCN degenerates into traditional GCN if the diagonal parameter $\alpha = 0$. To explore the difference between GCN and type-1 first-order TGCN, we adjust the parameter α both manually and automatically. The overall accuracy is reported in Table 6.3.

For type-1 first-order TGCN, our manual adjustment results in higher accuracy than automatic adjustment. This indicates that the deep learning network may be more susceptible to local convergence when learning α by itself. Usually, the optimal α for type-1 TGCN would be in $[0.15, 0.35]$ as shown in Fig. 6.1, while the TGCN degenerates to the traditional GCN for $\alpha = 0$. Generally, type-2 first-order TGCN has a clear advantage in accuracy for all datasets, whereas type-1 frist-order TGCN exhibits only marginal improvement given

Table 6.4: Accuracy for Higher-Order Propagation Model (Percent)

Num of Layers	Polynomial Order k	TGCN Type	Cora	Citeseer
2-Layer	1^{st}-order	Type-1	81.4	70.1
2-Layer	1^{st}-order	Type-2	**81.5**	**70.5**
2-Layer	2^{nd}-order	Type-3	79.5	65.7
2-Layer	2^{nd}-order	Type-4	79.3	66.9
1-Layer	1^{st}-order	Type-1	75.6	67.3
1-Layer	2^{nd}-order	Type-3	78.4	69.0
1-Layer	3^{rd}-order	Type-3	79.1	68.4
1-Layer	2^{nd}-order	Type-4	76.3	67.9
1-Layer	3^{rd}-order	Type-4	78.6	70.2

* For each method, we test with the representing matrix $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$.

suitable choice of diagonal parameters.

We then compare different propagation models with different orders of polynomials under the same experiment setup. We start with 100 Monte Carlo random initializations and report the average accuracy of each model in Table 6.4. The first-order TGCNs generally achieve superior overall accuracy than higher-order TGCNs. Higher-order methods may be occasionally better for some datasets. Recall that the multi-layer first-order TGCN is a special case of single-layer higher-order polynomials as illustrated in Section 6.4.2. The large number of parameters in the higher-order methods may leads to highly likelihood of overfitting and local convergence, thereby contributing to their less impressive outcomes.

Depth and Polynomial Orders

We also test the effects of different polynomial orders and layer numbers. The accuracy and training time (200 epochs) for different polynomial orders (Type-3 as an example) are shown in the first group of plots in Fig. 6.2. We note that performance improvement appears to saturate beyond certain polynomial order. Since the results also indicate growing training time for higher order polynomials, it would be more efficient to limit the polynomial order to 2 or 3. We also test the performance of first-order TGCNs (Type-2 as an example) with different layers in the last two plots in Fig. 6.2, which also show that a 2-layer or 3-layer TGCN would typically suffice.

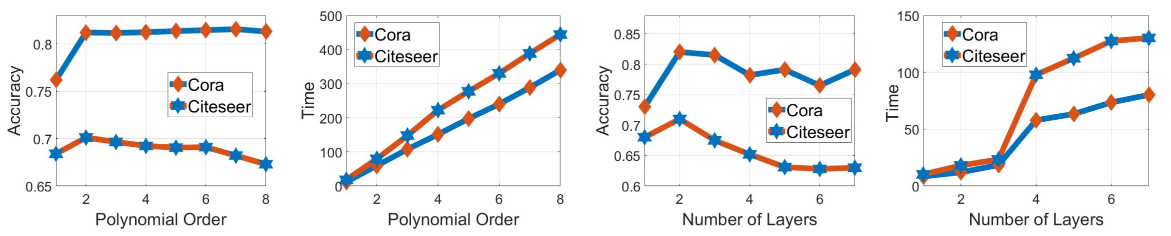
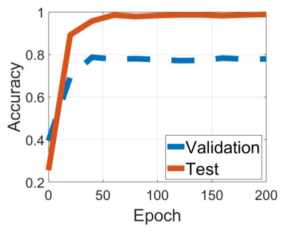
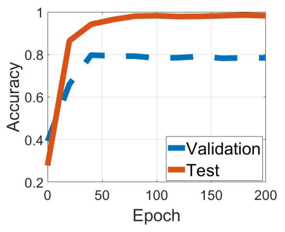


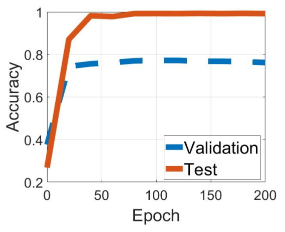
Figure 6.2: Results of Different Polynomial Orders and Network Depth.



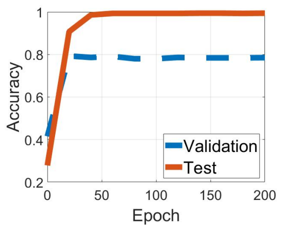
(a) 2-Layer Type-1 TGCN.



(b) 2-Layer Type-2 TGCN.



(c) 1-Layer 2-order Type-3 TGCN.



(d) 1-Layer 2-order Type-4 TGCN.

Figure 6.3: Convergence of different TGCN models.

Convergence

We evaluate the convergence of different TGCN models in Fig. 6.3. Here, we report the accuracy of training data and validation data for the Cora dataset. Form the results, we can see that TGCN models can converge well in the citation network datasets.

Training Efficiency

We compare the training efficiency for different methods based on the average training time for each epoch over 200 epochs in total. We use the same number of hidden units for multi-layer graph convolutional networks to be fair. From the results of Table 6.5, 2-layer TGCN takes nearly 10% longer than traditional 2-layer GCN because of the larger number of parameters and matrix computations. Moreover, larger layer depth and higher polynomial order also increase TGCN training time.

Table 6.5: Training Time per Epoch

Dataset	GCN	2L1KT1	2L1KT2	2L2KT3	2L2KT4
Cora	21.2ms	24.3ms	34.0ms	506.1ms	479.5ms
Citesser	30.5ms	33.4ms	42.9ms	681.1ms	726.5ms
Dataset	1L2KT3	1L3KT3	1L2KT4	1L3KT4	1L1KT1
Cora	253.8ms	463.2ms	244.5ms	641.0ms	11.3ms
Citesser	339.3ms	609.7ms	314.7ms	1085.6ms	33.2ms

* Different methods are measured in a CPU-only implementation.

* aLbKTc is short for a Type-c TGCN with a layers and polynomial order $k = b$.

Different Choices of Graph Representations

Thanks to the flexibility of $\mathbf{P}$ in the TGCN, it is interesting to explore different graph representations in different types of TGCN propagation models. Note that the Laplacian-based model can be written in the form of the adjacency matrix and a corresponding diagonal matrix, which can be included within the category of adjacency-based convolutional propagation models in TGCNs. Thus, we mainly investigate adjacency-based representation for TGCNs. Since there is no constraint on the range of eigenvalues, we test the effect of normalization. More specifically, we use the original adjacency matrix $\mathbf{A}$, the normalized adjacency matrix $\mathbf{D}^{-1}\mathbf{A}$, the traditional GCN propagation $\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$, and the pagerank $0.1 \times (\mathbf{I} - 0.9\tilde{\mathbf{A}})^{-1}$ [169]. For the higher-order TGCNs, we mainly focus on Cora and Citeseer datasets due to complexity.

The experiment results are shown in Table 6.6. The results show that normalized representations exhibit better performance than the unnormalized graph representation for TGCN propagation. More specifically, the normalized adjacency matrix achieves better performance in most TGCN designs. Although the pagerank propagation also shows good performance in some of the TGCN categories, the high-complexity of calculating the inverse matrix is detrimental to its applications in higher-order TGCNs. Note that we only test some of the common graph representations of $\mathbf{P}$ in our TGCN designs. The Taylor expansions allow a more flexible combination with other existing GCN propagations, such as pagerank

Table 6.6: Performance of Different Graph Representations

Dataset	Cora	Citeseer	Pubmed
Type-1 TGCN (2-Layer/Auto-training on α)			
$\mathbf{A}$	76.0	67.8	77.9
$\mathbf{D}^{-1}\mathbf{A}$	79.3	**70.2**	**80.5**
$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$	78.6	69.2	78.9
$0.1 \times (\mathbf{I} - 0.9\tilde{\mathbf{A}})^{-1}$	**80.1**	70.1	79.3
Type-2 TGCN (2-Layer)			
$\mathbf{A}$	76.8	67.6	77.8
$\mathbf{D}^{-1}\mathbf{A}$	**81.9**	70.0	**80.3**
$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$	81.6	70.1	79.4
$0.1 \times (\mathbf{I} - 0.9\tilde{\mathbf{A}})^{-1}$	81.8	**70.1**	80.1
Type-3 TGCN (1-Layer 2^{nd}-Order)			
$\mathbf{A}$	77.9	68.5	/
$\mathbf{D}^{-1}\mathbf{A}$	**79.0**	68.9	/
$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$	78.5	**69.1**	/
$0.1 \times (\mathbf{I} - 0.9\tilde{\mathbf{A}})^{-1}$	78.8	68.7	/
Type-4 TGCN (1-Layer 2^{nd}-Order)			
$\mathbf{A}$	75.2	67.0	/
$\mathbf{D}^{-1}\mathbf{A}$	**79.7**	**68.2**	/
$\tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$	76.3	67.9	/
$0.1 \times (\mathbf{I} - 0.9\tilde{\mathbf{A}})^{-1}$	78.3	68.0	/

and GCN propagations. We plan to further investigate alternative graph representations in future works.

Discussion

In terms of formulation, type-1 TGCN is an extension of GCN, which allows flexible self-influence for each node. Type-2 TGCN is an extension of type-1 TGCN, where different self-influence parameters are assigned for different nodes. In practical applications, such self-influence does exist and may be less obvious. Type-2 TGCN allows different self-influence parameters to be learned while training, which may lead to better performance in the citation networks. Higher-order TGCNs, as discussed in Section 6.4.2, are different from multi-layer TGCN as various orders may lead to different performances. However, to mitigate complexity

Table 6.7: Comparison with Other Methods in the Citation Networks

	Cora	Citeseer	Pubmed
GCN	85.77	73.68	88.13
GAT	86.37	74.32	87.62
GIN	86.20	76.80	87.39
Geom-GCN-I	85.19	77.99	**90.05**
Geom-GCN-P	84.93	75.14	88.09
Geom-GCN-S	85.27	74.71	84.75
APPNP	86.88	77.74	88.41
Type-1 TGCN	86.79	77.82	87.99
Type-2 TGCN	**87.23**	**78.31**	86.89

concerns, lower-order TGCNs are more efficient in applications. With the steady advances of computation hardware, higher-order TGCNs is expected to play increasingly important roles in future data analysis. In addition, the TGCN designs show a scalable combination with existing GCN propagations and graph representations.

6.6.2 Comparison with Several Existing Methods

In this section, we compare our proposed TGCNs with several state-of-the-art methods in two different tasks: 1) node classification in the citation network; and 2) point cloud segmentation.

Classification in the Citation Networks

We first compare the proposed TGCN frameworks with other GCN-style methods in the citation networks summarized in Table 6.2, but with a different splits on the datasets. Instead of randomly splitting a subset of the citation networks, we apply similar data splits as [178] with 60%/20%/20% for training/testing/validation datasets. We compare our methods with graph convolutional networks (GCN) [119], geometric graph convolutional networks (Geom-GCN) [178], graph attention networks (GAT) [176], approximated personalized propagation of neural predictions (APPNP) [169], and graph isomorphism networks (GIN) [181]. In TGCN propagation, we use the normalized adjacency matrix, i.e., $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$, and set the

Table 6.8: Mean Accuracy in ShapeNet Dataset.

	Type-2 TGCN[1]	Type-1 TGCN[1]	Type-2 TGCN[2]	Type-1 TGCN[2]	GSP	HGSP	GCN	PointNet
Airplane	0.7551	**0.7883**	0.7785	0.7824	0.5272	0.5566	0.7660	*0.834*
Bag	0.9165	0.9202	0.9399	***0.9409***	0.5942	0.5620	0.9176	0.787
Cap	**0.7670**	0.7599	0.7479	0.7577	0.6698	0.7212	0.7629	*0.825*
Car	**0.7114**	0.7052	0.7018	0.6976	0.3785	0.3702	0.6790	*0.749*
Chair	0.6603	0.6197	**0.7412**	0.6885	0.4701	0.5782	0.6430	*0.896*
Earphone	0.7037	0.7135	0.7606	***0.7712***	0.4706	0.5637	0.7054	0.730
Guitar	**0.8449**	0.8401	0.8176	0.8265	0.5731	0.5889	0.8304	*0.915*
Knife	**0.7675**	0.7474	0.7610	0.7614	0.6395	0.7045	0.7502	*0.859*
Lamp	0.7787	0.7836	**0.7853**	0.7712	0.2510	0.3112	0.7821	*0.808*
Laptop	0.8142	0.8365	0.8185	0.8275	0.6704	**0.9077**	0.8272	*0.953*
Motorbike	0.7167	0.7183	0.7239	0.7623	***0.7663***	0.7588	0.7297	0.652
Mug	0.9324	***0.9436***	0.9348	0.9376	0.7465	0.6290	0.9302	0.930
Pistol	0.7362	**0.7387**	0.7145	0.7107	0.5336	0.6277	0.7205	*0.812*
Rocket	***0.7895***	0.7712	0.7824	0.7742	0.4792	0.5481	0.7807	0.579
Skateboard	0.8323	0.8364	0.8230	***0.8493***	0.6088	0.5440	0.8176	0.728
Table	0.7984	0.8154	0.7972	0.*8194*	0.4726	0.4568	0.8164	0.806
Mean	0.7828	0.7836	0.7892	**0.7966**	0.5532	0.5893	0.7788	*0.803*

[1] TGCN[1] applies the graph representation $\mathbf{P} = \tilde{\mathbf{D}}^{-\frac{1}{2}}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-\frac{1}{2}}$.
[2] TGCN[2] applies the graph representation $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$.
[*] The best graph-based method is marked in bold font.
[*] The best method is marked in italic script.

number of layer to be two. The test results are reported in Table 6.7. The results show that the proposed TGCN achieves competitive performance against various GCN-type methods. Together with the results presented in Section 6.6.1, our experiments demonstrate that TGCN is very effective in node classification over the citation graphs despite data splits.

Point Cloud Segmentation

We next test the performance of TGCN in the point cloud segmentation. The goal of point cloud segmentation is to identify and cluster points in a point cloud that share similar features into their respective regions [161]. The segmentation problem can be posed as a semi-supervised classification problem if the labels of several samples are known [190].

Datasets and Baselines: In this work, we use the ShapeNet datasets [156, 157] as examples. In this dataset, there are 16 object categories, each of which may contain 2-6 classes. We compare both type-1 and type-2 TGCNs with traditional GCN in all categories. To explore the features extracted from the graph convolution, we also compare the proposed methods with geometric clustering-based methods, including the graph spectral clustering (GSP) and hypergraph spectral clustering (HGSP) [191]. A comparison with a specifically-

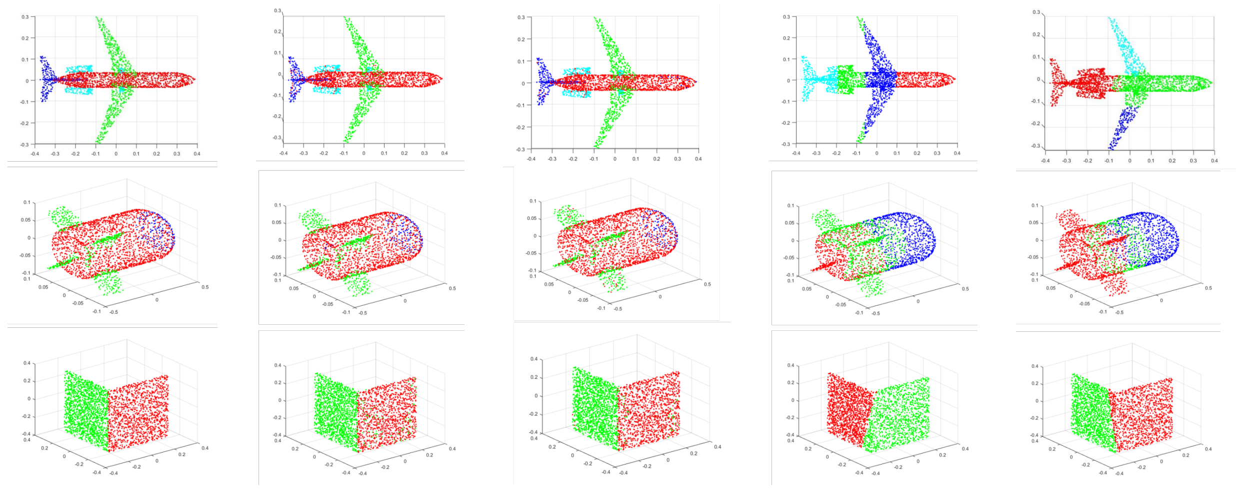

(a) Ground Truth. (b) TGCN. (c) GCN. (d) GSP. (e) HGSP.

Figure 6.4: Examples of Point Cloud Segmentation.

designed neural networks for point cloud segmentation, i. e., PointNet [192], is also reported.

Experiment Setup:

- To implement TGCN efficiently, we randomly pick 20 point cloud objects from each category, and randomly set 70% points as training data with labels while using the remaining points as the test data for each point cloud. We use k-nearest neighbor method to construct an adjacency matrix $\mathbf{A}$ with elements $a_{ij} = 1,\ 0$ to indicate the presence or the absence of connection between two nodes i, j, respectively. More specifically, we set $k = 20$ in graph construction for all point clouds. For the GCN-like methods, we fix the number of layer as two and the number hidden units as 40 to ensure a fair comparison.

- For the spectral-clustering based methods, we use the hypergraph stationary process [168] to estimate the hypergraph spectrum for the HGSP-based method, and apply the Gaussian distance model [135] to construct the graph for the GSP-based method. The k-means clustering is applied for segmentation after obtaining the key spectra.

Experiment Results: The overall accuracies of different methods are reported in Table 6.8. The resulting mean accuracy of segmentation illustrates that each method under comparison may exhibit some unique strength in different categories. The PointNet exhibits an overall largest mean accuracy, since it is specifically designed for point cloud segmentation while the GCN-like methods are directly applied in classifying the points without adjustment. Even though, the TGCN still shows better performances in some of the categories than PointNet, such as table, skateboard, mug, motorbike, earphone and bag. Compared to the traditional GCN, TGCN generally achieves higher accuracy and provides the better performance in most of the categories. For the graph representation, the normalized adjacency matrix performs better than the traditional GCN propagation matrix in the TGCN designs for point cloud segmentation. The clustering-based methods perform worse than the classification-based methods, since they use no prior knowledge of the ground truth. However, they still achieve satisfying performances in the point clouds with fewer classes and more regular shapes.

To further illustrate different methods, several visualized segmentation results are presented in Fig. 6.4. Since different TGCN exhibits similar visualized results, we report type-2 TGCN with $\mathbf{P} = \mathbf{D}^{-1}\mathbf{A}$ as an example. From the results, we see that the classification-based methods, i. e., TGCN and GCN, exhibit similar results, where errors are distributed scatteredly over the point clouds. Generally, the TGCN results show fewer errors than those of the traditional GCN, such as in the wings of rockets and planes. Different from GCN-like methods trained according to the ground truth, the clustering-based methods show different results. For example, in the first row of Fig. 6.4(e), although the segmentation result differ from the ground truth, these results still make sense by grouping two wings to different classes. In addition, the errors of clustering-based methods are grouped together in the intersections of two classes. It will be interesting to explore the integration of classification-based methods and clustering-based methods to extract more features of point clouds in the future works.

6.7 Conclusions

In this chapter, we explore the inherent connection between GSP convolutional spectrum wavelet and the GCN vertex propagation. Our analysis shows that spectral wavelet-kernel can be approximated in vertex domain if it admits a polynomial approximation. In addition, we develop an efficient and simple alternative design of GCN layers based on the simple Taylor expansion (TGCN), which exhibits computation efficiency and outperforms a number of state-of-the-art GCN-type methods. We also derive practical guidelines on the selection of representing matrix and the propagation model for TGCN designs. Our interpretability discussion presents good insights into Taylor approximation of graph convolution.

Chapter 7

Conclusions and Future Direction

In this chapter, we summarize our contributions in this dissertation and suggest several future directions.

7.1 Summary and Conclusions

In this dissertation, we investigate the propagation analysis in the high dimensional networks, and explore the design of novel signal processing and learning tools based on graphs. These approaches aim to capture the underlying correlations between data samples or object nodes to benefit the analysis and processing in real scenarios.

To tackle the order limitations of matrix representation, we venture into the high-dimensional tensor algebra in Chapter 2. More specifically, we propose tensor-based methods to model the multilayer networks and hypergraphs, and develop spectral analysis over the tensor models. With deeper understanding of the properties in high-dimensional graphs, we come back to normal graphs to improve the design of graph learning machines. The detailed research topics addressed in this dissertation are summarized as follows:

- In chapter 3, we propose a scalable tensor analysis for the failure propagation over multilayer networks. Specifically, we provide a spectral analysis of epidemic spread based on the SIS model over the multilayer complex systems. To capture the propagation property to measure the system stability, the failure transition equation and failure threshold are derived. For the purpose of computation efficiency, we derive the upper and lower bounds for the failure indicator, as well as its approximations in special cases. Our bounds are shown to be tight, and the approximation are close to the exact values in the numerical analysis.

- In chapter 4, we propose a novel framework of Hypergraph Signal Processing (HGSP) to generalize the traditional GSP to high-dimensional hypergraphs. We provide the theoretical fundamentals of HGSP, including hypergraph signal, hypergraph shifting, hypergraph frequency and bandlimited signals. We then introduce the basic HGSP tools, such as sampling theory and filtering design. Our experiment results in several application examples validate the advantage of the proposed HGSP framework and the power of hypergraph in capturing high-dimensional correlations.

- In chapter 5, we investigate the applications of HGSP tools in multimedia processing. We first introduce some novel HGSP operations, which are useful for multimedia analysis, including convolution, translation, sampling and stationary process. Then, we present different methods to construct hypergraph models for images, videos and point clouds, respectively. Within the proposed framework, HGSP is successfully applied in the area of image compression, video segmentation, edge detection, point cloud resampling and denoising, according to the experimental results in both synthetic and real datasets.

- In chapter 6, we explore the inherent connections between GSP and Graph Convolutional Networks. We show that the spectral wavelet-kernel can be approximated by vertex propagation if it admits a polynomial approximations. Then, we introduce the alternative design fo GCN layers based on the Taylor expansions. The proposed framework exhibits computation efficiency and outperforms a number of state-of-art

GCN-type methods, which validate the advantage of the processed TGCN.

7.2 Future Directions

Signal processing and learning over graph are now drawing increasing attentions. This new and highpotential subject still has many research problems that are worth of investigating. We briefly present several future directions as follows.

- **Efficient Implementation of HGSP Operations**: Although the HGSP framework has shown significant benefits in a number of applications, it still suffer from the implementation inefficacy. On the one hand, with the increase of hyperedge dimension and number of data samples, the size of representing tensor can be large and lack of storage efficacy. On the other hand, the calculation of hypergraph spectrum based on tensor decomposition is time-consuming for the large datasets. To accommodate HGSP to the requirement big data, there are several potential future directions to further improve it. The first is the sparse representation of hypergraph representing tensor. As aforementioned in Section 5.4.2, a realistic network and graph models are sparse. Thus, it will benefit the application of HGSP if sparse analysis can be integrated with HGSP. Second, similar to what we do in Section 5.4 and 6.4, approximated methods to estimate the hypergraph spectrum and spectral operations based on properties in the vertex domain are worth further investigations to reduce complexity. Other aspects, such as fast hypergraph Fourier transform and fast HGSP filter implementation, can be also promising future directions.

- **Development of Hypergraph Convolutional Networks**: Inspire by the design of GCNs [119], matrix-based hypergraph convolutional networks are developed to extract high-dimensional features of the datasets [25,122]. However, the matrix-based method can be interpreted as a special case of GCNs, where the hypergraph similarity matrix is another construction of graph models. In addition, the matrix-based hypergraph approximation lacks the definition of convolution in the signal processing viewpoints. To this end, it will be interesting to explore the development of tensor-based hypergraph convolutional networks. The tensor-based HGSP provides an alternation definition of hypergraph convolution, and is an intuitive extension of traditional graph convolution. Moreover, the tensor-based representation highlights the cross-feature between data samples, which is different from traditional GCNs.

- **Signal Processing over Dynamic Graphs**: Traditional graph signal processing and hypergraph signal processing mainly focus the propagation of the signals over a static graph/hypergraph. Once the graph structure changes, the graph spectrum needs to be recalculated, which is less efficient in real applications. Development of signal processing over dynamic graph could be an urgent research direction in the future. One method is to model the dynamic graph as a spatial-temporal graph [193], which can be represented by multilayer networks or product graphs, for further analysis.

- **Design of Graph/Hypergraph Auto-Encoder**: With increasing attentions in graph learning, the development of graph/hypergraph auto-encoder (GAE) is another interesting future direction. There are mainly two directions to design a GAE. The first is the variational GAE, which integrate GCN in the traditional variational auto-encoder to embed the graph structure [194]. The other one is based on the subgraph encoding [195], which encodes the original graph into low-dimensional subgraphs. In addition, the integration of GAE with traditional subspace learning, such as rate reduction [196], can be also promising.

Appendix A

Proofs for Chapter 3

A.1 Proof of Lemma 3.2

Proof. Suppose that λ is an eigenvalue of $\mathbf{A}$, then we have

$$det(\mathbf{A} - \lambda\mathbf{I}) = 0. \tag{A.1}$$

Since

$$det(\mathbf{A} - \lambda\mathbf{I}) = \prod_{i=1}^{K} det(\mathbf{B_i} - \lambda\mathbf{I_i}), \tag{A.2}$$

the eigenvalues of $\mathbf{A}$ is the list of the eigenvalues of all the $\mathbf{B_i}$. Then, we can easily show that the largest eigenvalue of $\mathbf{A}$ is the largest eigenvalue of $\mathbf{B_i}$. $\square$

A.2 Proof of Lemma 3.3

Proof. Suppose that the adjacency tensor $\mathbf{S}$ has an eigenvalue λ, i.e.,

$$\mathbf{S}\mathbf{x} = \lambda\mathbf{x}, \tag{A.3}$$

where $\mathbf{x} = [\mathbf{x}_{\hat{1},N}^\top, \mathbf{x}_{\hat{2},N}^\top, \cdots, \mathbf{x}_{\hat{M},N}^\top]^\top$ is a the corresponding eigenvector associated with λ.

Then, we obtain

$$\sum_{\hat{i}\neq\hat{\beta}} w_{\hat{i}}^{\hat{\beta}} \mathbf{I_N} \mathbf{x}_{\hat{i},N} = \sum_{\hat{i}\neq\hat{\beta}} w_{\hat{i}}^{\hat{\beta}} \mathbf{x}_{\hat{i},N} = \lambda \mathbf{x}_{\hat{\beta},N}. \tag{A.4}$$

We can pick the first entry in each vector $\mathbf{x}_{\hat{i},N}$, denoted by $a_{\hat{i}}$. Then, we have

$$\sum_{\hat{i}\neq\hat{\beta}} w_{\hat{i}}^{\hat{\beta}} a_{\hat{i}} = \lambda a_{\hat{\beta}}. \tag{A.5}$$

Now, we construct a vector $\mathbf{y} = [a_1, \cdots, a_N]^\top$. According to Eq. (A.5), we have

$$\mathbf{W}\mathbf{y} = \lambda \mathbf{y}. \tag{A.6}$$

Thus, λ is also an eigenvalue of $\mathbf{W}$. Since λ can be any eigenvalue of $\mathbf{S}$, all the eigenvalues of $\mathbf{S}$ are the eigenvalues of $\mathbf{W}$. Now, let's prove that all the eigenvalues of $\mathbf{W}$ are the eigenvalues of $\mathbf{S}$. Suppose that $\mathbf{W}$ has an eigenvalue λ and the corresponding eigenvector $\mathbf{y}$. Then, it should satisfy Eq. (A.6). Constructing $\mathbf{x}_{\hat{i},N}$ in Eq. (A.4) by duplicating ith element in $\mathbf{y}$, we can easily obtain a vector $\mathbf{x} = [\mathbf{x}_{\hat{1},N}^\top, \mathbf{x}_{\hat{2},N}^\top, \cdots, \mathbf{x}_{\hat{M},N}^\top]^\top$, which makes Eq. (A.3) exist, i.e., λ is also the eigenvalue of $\mathbf{S}$. Then, we can show that all the eigenvalues of $\mathbf{W}$ are the eigenvalues of $\mathbf{S}$. □

A.3 Proof of Theorem 3.1

Proof. Let $\mathbf{U} \in \mathbb{R}^{M\times M\times N\times N}$ be a forth-order tensor with elements $U_{i,\hat{\alpha}}^{j,\hat{\beta}} = (1-\mu_{\hat{\alpha}})\delta_{i,\hat{\alpha}}^{j,\hat{\beta}}$ and $\boldsymbol{\theta} \in \mathbb{R}^{M\times M\times N\times N}$ be a 4-order tensor with elements $\theta_{i,\hat{\alpha}}^{j,\hat{\beta}}$. Then, we can write the $MN \times MN$ flattened transition tensor as

$$\mathbf{T} = \mathbf{U} + \boldsymbol{\theta} * \mathbf{A} = \mathbf{U} + \boldsymbol{\theta} * \mathbf{A}_{intra} + \boldsymbol{\theta} * \mathbf{A}_{inter}, \tag{A.7}$$

where $\mathbf{A}_{intra} = diag([\mathbf{A}_{\hat{\mathbf{1}}}^{\hat{\mathbf{1}}} \quad ... \quad \mathbf{A}_{\hat{\mathbf{M}}}^{\hat{\mathbf{M}}}])$ and $\mathbf{A}_{inter} = \mathbf{A} - \mathbf{A}_{intra}$. With the property of spectral radius, we have

$$\rho(\mathbf{T}) \leq \rho(\mathbf{U}) + \rho(\boldsymbol{\theta} * \mathbf{A}_{inter}) + \rho(\boldsymbol{\theta} * \mathbf{A}_{intra}). \tag{A.8}$$

Let us analyze each term in Eq. (A.8):

- $\rho(\mathbf{U})$: $\mathbf{U} = diag([(1-\mu_{\hat{1}})\mathbf{I} \quad ... \quad (1-\mu_{\hat{M}})\mathbf{I}])$.

  According to *Lemma 3.2*, we have

$$\rho(\mathbf{U}) = \max_{\hat{\alpha}} \rho[(1-\mu_{\hat{\alpha}})\mathbf{I}] = 1 - \min_{\hat{\alpha}}\{\mu_{\hat{\alpha}}\}. \tag{A.9}$$

- $\rho(\boldsymbol{\theta} * \mathbf{A}_{intra})$: According to *Lemma 3.2*, we have the following result.

$$\begin{aligned} \rho(\boldsymbol{\theta} * \mathbf{A}_{intra}) &= \rho(diag([\theta_{\hat{1}}^{\hat{1}}\mathbf{A}_{\hat{\mathbf{1}}}^{\hat{\mathbf{1}}} \quad ... \quad \theta_{\hat{M}}^{\hat{M}}\mathbf{A}_{\hat{\mathbf{M}}}^{\hat{\mathbf{M}}}])) & \text{(A.10)} \\ &= \max_{\hat{\alpha}}\{\rho(\theta_{\hat{\alpha}}^{\hat{\alpha}}\mathbf{A}_{\hat{\alpha}}^{\hat{\alpha}})\}. & \text{(A.11)} \end{aligned}$$

- $\rho(\boldsymbol{\theta} * \mathbf{A}_{inter})$:

$$\rho(\boldsymbol{\theta} * \mathbf{A}_{inter}) = \rho\left(\begin{bmatrix} 0 & \cdots & \theta_{\hat{M}}^{\hat{1}}\mathbf{A}_{\hat{\mathbf{M}}}^{\hat{\mathbf{1}}} \\ \vdots & \ddots & \vdots \\ \theta_{\hat{1}}^{\hat{M}}\mathbf{A}_{\hat{\mathbf{1}}}^{\hat{\mathbf{M}}} & \cdots & 0 \end{bmatrix}\right). \tag{A.12}$$

As we mentioned in the third property of the tensor framework in Section 3.3, $\mathbf{A}_{\hat{\alpha}}^{\hat{\beta}}$ is diagonal, i.e., $\mathbf{A}_{\hat{\alpha}}^{\hat{\beta}} = diag\{\mathbf{v}_{\hat{\beta},\hat{\alpha}}\}$, where $\mathbf{v}_{\hat{\beta},\hat{\alpha}}$ is a N-length vector composed of 1 and 0. Then, we can construct another diagonal matrix $\mathbf{Q}_{\hat{\alpha}}^{\hat{\beta}}$, which satisfies $\theta_{\hat{\alpha}}^{\hat{\beta}}\mathbf{A}_{\hat{\alpha}}^{\hat{\beta}} + \mathbf{Q}_{\hat{\alpha}}^{\hat{\beta}} = \theta_{\hat{\alpha}}^{\hat{\beta}}\mathbf{I}_{\mathbf{N}}$.

Define

$$\mathbf{Z} = \begin{bmatrix} 0 & \mathbf{Q}_{\hat{\mathbf{2}}}^{\hat{\mathbf{1}}} & \cdots & \mathbf{Q}_{\hat{\mathbf{M}}}^{\hat{\mathbf{1}}} \\ \mathbf{Q}_{\hat{\mathbf{1}}}^{\hat{\mathbf{2}}} & 0 & \cdots & \mathbf{Q}_{\hat{\mathbf{M}}}^{\hat{\mathbf{2}}} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{Q}_{\hat{\mathbf{1}}}^{\hat{\mathbf{M}}} & \mathbf{Q}_{\hat{\mathbf{2}}}^{\hat{\mathbf{M}}} & \cdots & 0 \end{bmatrix}. \tag{A.13}$$

The result of $\mathbf{Z} + \boldsymbol{\theta} * \mathbf{A}_{inter}$ is the same as the inter-layer adjacency tensor $\mathbf{S}$ defined in *Lemma 3.3*. The infection rates $\boldsymbol{\theta}$ here can be seen as the weights $\mathbf{W}$. Then,

$$\mathbf{S} = \mathbf{Z} + \boldsymbol{\theta} * \mathbf{A}_{inter}. \tag{A.14}$$

According to *Lemma 3.1*, we obtain

$$\rho(\mathbf{S}) \geq \max\{\rho(\mathbf{Z}), \rho(\boldsymbol{\theta} * \mathbf{A}_{inter})\} \geq \rho(\boldsymbol{\theta} * \mathbf{A}_{inter}). \tag{A.15}$$

As we proved in *Lemma 3.3*, $\rho(\mathbf{S}) = \rho(\hat{\boldsymbol{\theta}})$. Then,

$$\rho(\boldsymbol{\theta} * \mathbf{A}_{inter}) \leq \rho(\hat{\boldsymbol{\theta}}). \tag{A.16}$$

According to Eqs. (A.9), (A.11) and (A.16), we obtain that $\rho(\mathbf{T})$ in Eq. (A.8) is upper-bounded as

$$\rho(\mathbf{T}) \leq 1 - \min_{\hat{\alpha}}\{\mu_{\hat{\alpha}}\} + \max_{\hat{\alpha}}\{\rho(\theta_{\hat{\alpha}}^{\hat{\alpha}}\mathbf{A}_{\hat{\alpha}}^{\hat{\alpha}})\} + \rho(\hat{\boldsymbol{\theta}}). \tag{A.17}$$

□

A.4 Proof of Theorem 3.2

Proof. As we have shown $\mathbf{T} = \mathbf{U} + \boldsymbol{\theta} * \mathbf{A}_{intra} + \boldsymbol{\theta} * \mathbf{A}_{inter}$ in Appendix A.5, we could easily use *Lemma 3.1* to prove that

$$\rho(\mathbf{T}) \geq \max\{\rho(\mathbf{U}), \rho(\boldsymbol{\theta} * \mathbf{A}_{intra}), \rho(\boldsymbol{\theta} * \mathbf{A}_{inter})\}. \tag{A.18}$$

For $\boldsymbol{\theta} * \mathbf{A}_{inter}$, it can be seen as the adjacency tensor of a multilayer network, which only has inter-layer connections. With the $NM \times NM$ flattening method, we can reshape the tensor as $\boldsymbol{\theta} * \mathbf{A}_{inter} = diag\{[\mathbf{D_1} \quad ... \quad \mathbf{D_N}]\}$, where $\mathbf{D_i} = \hat{\boldsymbol{\theta}}_{\boldsymbol{j}}$ for $\sum_{k=1}^{k=j-1} N_k < i \leq \sum_{k=1}^{k=j} N_k$. Then, with *Lemma 3.2*, we have

$$\rho(\boldsymbol{\theta} * \mathbf{A}_{inter}) = \max_i\{\rho(\mathbf{D_i})\} = \max_j\{\rho(\hat{\boldsymbol{\theta}}_{\boldsymbol{j}})\}. \tag{A.19}$$

According to Eqs. (A.9), (A.11) and (A.19), we obtain that $\rho(\mathbf{T})$ in Eq. (A.18) as lower-bounded as

$$\rho(\mathbf{T}) \geq \max_{i,\hat{\alpha}}\{1 - \mu_\alpha, \rho(\theta^{\hat{\alpha}}_{\hat{\alpha}} \mathbf{A}^{\hat{\alpha}}_{\hat{\alpha}}), \rho(\hat{\boldsymbol{\theta}}_{\boldsymbol{i}})\}. \tag{A.20}$$

□

A.5 Proof of Theorem 3.3

Proof. Let $\theta_m = \max_{\hat{\alpha}}\{\theta^{\hat{\alpha}}_{\hat{\alpha}}\}$. As $\theta^{\hat{\alpha}}_{\hat{\alpha}} \gg \theta^{\hat{\alpha}}_{\hat{\beta}}$, we have $\frac{\theta^{\hat{\alpha}}_{\hat{\beta}}}{\theta_m} \approx 0$ for any $\hat{\alpha}$ and $\hat{\beta}$. With *Lemma 3.2*, we have

$$
\begin{aligned}
&\rho(\mathbf{T}) \\
&\approx \theta_m \rho \left(\begin{bmatrix} \frac{(1-\mu_{\hat{1}})}{\theta_m}\mathbf{I_N} + \frac{\theta^{\hat{1}}_{\hat{1}}}{\theta_m}\mathbf{A}^{\hat{\mathbf{1}}}_{\hat{\mathbf{1}}} & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \frac{(1-\mu_{\hat{M}})}{\theta_m}\mathbf{I_N} + \frac{\theta^{\hat{M}}_{\hat{M}}}{\theta_m}\mathbf{A}^{\hat{\mathbf{M}}}_{\hat{\mathbf{M}}} \end{bmatrix} \right) \\
&= \max_{\hat{\alpha}}\{1-\mu_{\hat{\alpha}} + \theta^{\hat{\alpha}}_{\hat{\alpha}}\rho(\mathbf{A}^{\alpha}_{\alpha})\}. \qquad (A.21)
\end{aligned}
$$

□

A.6 Proof of Theorem 3.4

Proof. Similar to Appendix A.5, we have $\frac{\theta^{\hat{\alpha}}_{\hat{\alpha}}}{\theta_m} \approx 0$ for $\theta^{\hat{\alpha}}_{\hat{\alpha}} \ll \theta^{\hat{\alpha}}_{\hat{\beta}}$ where $\theta_m = \max_{\hat{\alpha},\hat{\beta}}\{\theta^{\hat{\alpha}}_{\hat{\beta}}\}$. Reshaping the $MN \times MN$ flattened tensor to the $NM \times NM$ flattened tensor,

$$\rho(\mathbf{T}) \approx \theta_m \rho[diag(\frac{1}{\theta_m}\boldsymbol{\hat{\theta}_i} + \frac{1}{\theta_m}\boldsymbol{\phi I_M})], \qquad (A.22)$$

where $\boldsymbol{\phi} = [1-\mu_{\hat{1}}, \cdots, 1-\mu_{\hat{M}}]$ and $\boldsymbol{\hat{\theta}_i} = \boldsymbol{\hat{\theta}} * \mathbf{A_{int}}(i)$.

We could easily prove the conclusion with *Lemma 3.2*. □

Appendix B

Proofs for Chapter 4

B.1 Proof of Theorem 4.1

Proof. For hypergraph signals, the output of one-time shifting of $\mathbf{f}_r$ is calculated as

$$\mathbf{f}_{r(1)} = \sum_{i=1}^{N} \lambda_i \mathbf{f}_i (\mathbf{f}_i^{\mathrm{T}} \mathbf{f}_r)^{M-1} = \lambda_r \mathbf{f}_r. \tag{B.1}$$

Based on the normalized $\mathbf{F}^{norm}$, we have $\mathbf{f}_{r(1)}^{norm} = \frac{\lambda_r}{\lambda_{max}} \mathbf{f}_r$. It is therefore easy to obtain Eq. (4.24c) from Eq. (4.24a). To obtain Eq. (4.24b), we have

$$\mathbf{P_s} \mathbf{f}_r = \sum_{i=1}^{N} \lambda_i \mathbf{f}_i (\mathbf{f}_i^{\mathrm{T}} \mathbf{f}_r) = \frac{\lambda_r}{\lambda_{max}} \mathbf{f}_r. \tag{B.2}$$

It is clear that Eq. (4.24b) is the same as Eq. (4.24c).

Since λ is real and nonnegative, we have

$$\mathbf{TV}(\mathbf{f}_i) - \mathbf{TV}(\mathbf{f}_j) = \frac{\lambda_j - \lambda_i}{\lambda_{max}}. \tag{B.3}$$

Obviously, $\mathbf{TV}(\mathbf{f}_i) > \mathbf{TV}(\mathbf{f}_j)$ iff $\lambda_i < \lambda_j$. $\square$

B.2 Proof of Theorem 4.2

Proof. We first examine one of the orders in n-Mode product of hypergraph signal, i.e., nth-order of $\mathbf{s}^{[\mathbf{M-1}]}$, $1 \leq n \leq N$, as

$$(\mathbf{s}^{[\mathbf{M-1}]} \times_n \mathbf{U})_{i_1 \ldots i_{n-1} j i_{n+1} \ldots i_{M-1}} = \sum_{i_n=1}^{N} s_{i_1} s_{i_2} \ldots s_{i_{M-1}} U_{j i_n}. \tag{B.4}$$

Since all elements in $\mathbf{s}_{\mathbf{Q}}^{[\mathbf{M-1}]}$ should also be the elements of $\mathbf{s}^{[\mathbf{M-1}]}$ after sampling, only one $U_{ji_n} = 1$ exists for each j according to Eq. (B.4), i.e., only one term in the summation exists for each j in the right part of Eq. (B.4). Moreover, since $\mathbf{U}$ samples over all the order, $U_{pi_n} = 1$ and $U_{ji_n} = 1$ cannot exist at the same time so that all the entries in $\mathbf{s}_{\mathbf{Q}}^{[\mathbf{M-1}]}$ are also in $\mathbf{s}^{[\mathbf{M-1}]}$. Suppose $q = \{q_1, q_2, \cdots, q_Q\}$ is the places of non-zero U_{jq_j}'s, we have

$$\mathbf{s}_{\mathbf{Q}}^{[\mathbf{M-1}]}(i_1, i_2, \cdots, i_Q) = s_{i_{q_1}} s_{i_{q_2}} \cdots s_{i_{q_Q}}. \tag{B.5}$$

As a result, we have $U_{ji} = \delta[i - q_j]$, which is the same as the sampling operator for the original signal. For the interpolation operator, the proof is similar and hence omitted. □

B.3 Proof of Lemma 4.1

Proof. Since $\mathbf{s}$ is K-bandlimited, $\tilde{\mathbf{s}}_i = \mathbf{f}_i^{\mathrm{T}} \mathbf{s} = 0$ when $i > K$. Then, according to Eq.(4.20), we have

$$\mathbf{s} = \mathbf{V}^{\mathrm{T}} \mathbf{V} \mathbf{s} = \sum_{i=1}^{K} \mathbf{f}_i \mathbf{f}_i^{\mathrm{T}} \mathbf{s} + \sum_{i=K+1}^{N} \mathbf{f}_i \mathbf{f}_i^{\mathrm{T}} \mathbf{s} = \sum_{i=1}^{K} \mathbf{f}_i \mathbf{f}_i^{\mathrm{T}} \mathbf{s} + 0 = \mathcal{F}_{[K]}^{\mathrm{T}} \tilde{\mathbf{s}}_{[K]}, \tag{B.6}$$

where $\mathbf{V} = [\mathbf{f}_1, \cdots, \mathbf{f}_N]^{\mathrm{T}}$. □

B.4 Proof of Theorem 4.3

Proof. To prove the theorem, we show that $\mathbf{TU}$ is a projection operator and $\mathbf{T}$ spans the space of the first K eigenvectors. From *Lemma 4.1* and $\mathbf{s} = \mathbf{T}\mathbf{s}_{\mathbf{Q}}$, we have

$$\mathbf{s} = \mathcal{F}_{[K]}^{\mathrm{T}} \tilde{\mathbf{s}}_{[K]} = \mathcal{F}_{[K]}^{\mathrm{T}} \mathbf{Z}\mathbf{s}_{\mathbf{Q}}. \tag{B.7}$$

As a result, $rank(\mathbf{Z}\mathbf{s}_{\mathbf{Q}}) = rank(\tilde{\mathbf{s}}_{[K]}) = K$. Hence, we conclude that $K \leq Q$.

Next, we show that $\mathbf{TU}$ is a projection by proving that $\mathbf{TU} \cdot \mathbf{TU} = \mathbf{TU}$. Since we have $Q \geq K$ and

$$\mathbf{ZU}\mathcal{F}_{[K]}^{\mathrm{T}} = \mathbf{I_K}, \tag{B.8}$$

We have

$$\mathbf{TU} \cdot \mathbf{TU} = \mathcal{F}_{[K]}^{\mathrm{T}} \mathbf{ZU} \mathcal{F}_{[K]}^{\mathrm{T}} \mathbf{ZU} \tag{B.9a}$$

$$= \mathcal{F}_{[K]}^{\mathrm{T}} \mathbf{ZU} = \mathbf{TU}. \tag{B.9b}$$

Hence, TU is a projection operator. For the spanning part, the proof is the same as that in [95]. □

B.5 Proof of Theorem 4.4

Proof. Let the diagonal matrix $\Sigma_{[K]}$ consist of the first K coefficients $\{\lambda_1, \ldots, \lambda_K\}$. Since $\mathbf{Z}\mathbf{U}\mathcal{F}_{[K]}^{\mathrm{T}} = \mathbf{I_K}$, we have

$$\begin{aligned}
\mathbf{F_K}\mathbf{s}_{[K]}^{[M-1]} &= \mathbf{Z}^{-1}\Sigma_{[K]}\hat{\mathbf{s}}_{[K]} && \text{(B.10a)}\\
&= \mathbf{U}\mathcal{F}_{[K]}^{\mathrm{T}}\Sigma_{[K]}\hat{\mathbf{s}}_{[K]} && \text{(B.10b)}\\
&= \mathbf{U}[(\sum_{i=1}^{K}\lambda_i \cdot \underbrace{\mathbf{f}_i \circ \ldots \circ \mathbf{f}_i}_{\text{M times}})(\underbrace{\mathbf{s} \circ \ldots \circ \mathbf{s}}_{\text{M-1 times}}) + 0] && \text{(B.10c)}\\
&= \mathbf{U}\mathbf{F}\mathbf{s}^{[M-1]}. && \text{(B.10d)}
\end{aligned}$$

Since $\mathbf{s}_{[K]} = \mathbf{U}\mathbf{s}$, it therefore holds that $\mathbf{s}_{[K]} - \mathbf{F_K}\mathbf{s}_{[K]}^{[M-1]} = \mathbf{U}(\mathbf{s} - \mathbf{F}\mathbf{s}^{[M-1]})$. □

B.6 Proof of Lemma 4.2

Proof. Let $\mathbf{V}^{\mathrm{T}} = [\mathbf{f}_1, \cdots, \mathbf{f}_N]$ and $\Sigma = diag([\lambda_1, \cdots, \lambda_N])$. Since $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$, we have

$$\begin{aligned}
\mathbf{P}^k &= \underbrace{\mathbf{V}^{\mathrm{T}}\Sigma\mathbf{V}\mathbf{V}^{\mathrm{T}}\Sigma\mathbf{V}\cdots\mathbf{V}^{\mathrm{T}}\Sigma\mathbf{V}}_{k \quad times} && \text{(B.11a)}\\
&= \mathbf{V}^{\mathrm{T}}\Sigma^k\mathbf{V}. && \text{(B.11b)}
\end{aligned}$$

Therefore, the kth-order term is given as

$$\begin{aligned}
\mathbf{s}_{<k>} &= \mathbf{V}^{\mathrm{T}}\Sigma^k\mathbf{V}\mathbf{s} && \text{(B.12a)}\\
&= \sum_{r=1}^{N}\lambda_r^k(\mathbf{f}_r^{\mathrm{T}}\mathbf{s})\mathbf{f}_r. && \text{(B.12b)}
\end{aligned}$$

□

Appendix C

Proofs for Chapter 5

C.1 Proof of Theorem 5.1

Proof. Since the hypergraph spectrum basis are orthonormal, we have $\mathbf{V}\mathbf{V}^T = \mathbf{I}$. Then, the τ-step shifting based on supporting matrix can be calculated as

$$\mathbf{P}_\tau = \underbrace{\mathbf{V}\Lambda_\mathbf{P}\mathbf{V}^T\mathbf{V}\Lambda_\mathbf{P}\mathbf{V}^T \cdots \mathbf{V}\Lambda_\mathbf{P}\mathbf{V}^T}_{\tau \quad times} \tag{C.1}$$

$$= \mathbf{V}\Lambda_\mathbf{P}^\tau\mathbf{V}^T. \tag{C.2}$$

Now, the Eq. (5.8) can be written as

$$\mathbb{E}[\mathbf{x}] = \mathbf{V}\Lambda_P^\tau\mathbf{V}^T\mathbb{E}[\mathbf{x}]. \tag{C.3}$$

Since $\mathbf{V}\Lambda_P^\tau\mathbf{V}^T$ does not always equal to $\mathbf{I}$, Eq. (5.8) holds for arbitrary supporting matrix and τ if and only if $\mathbb{E}[\mathbf{x}] = \mathbf{0}$.

Next we show the sufficiency and necessity of the condition in Eq. (5.11). The condition in Eq. (5.9) can be written as

$$\mathbf{P}_{\tau_1}\mathbb{E}[\mathbf{x}\mathbf{x}^H]((\mathbf{P})_{\tau_2}^H)^H = \mathbf{P}_{\tau_1+\tau}\mathbb{E}[\mathbf{x}\mathbf{x}^H]((\mathbf{P})_{\tau_2-\tau}^H)^H. \tag{C.4}$$

Considering Eq. (C.2) and the fact that hypergraph spectrum is real [123], Eq. (5.9) is equivalent to

$$\mathbf{V}\Lambda_{\mathbf{P}}^{\tau_1}\mathbf{V}^H\mathbb{E}[\mathbf{x}\mathbf{x}^H]\mathbf{V}\Lambda_{\mathbf{P}}^{\tau_2}\mathbf{V}^H = \mathbf{V}\Lambda_{\mathbf{P}}^{\tau_1+\tau}\mathbf{V}^H\mathbb{E}[\mathbf{x}\mathbf{x}^H]\mathbf{V}\Lambda_{\mathbf{P}}^{\tau_2-\tau}\mathbf{V}^H, \tag{C.5}$$

which can be written as

$$(\mathbf{V}^H\mathbb{E}[\mathbf{x}\mathbf{x}^H]\mathbf{V})\Lambda_{\mathbf{P}}^{\tau} = \Lambda_{\mathbf{P}}^{\tau}(\mathbf{V}^H\mathbb{E}[\mathbf{x}\mathbf{x}^H]\mathbf{V}). \tag{C.6}$$

If Eq. (C.6) holds for arbitrary $\mathbf{P}$, $(\mathbf{V}^H\mathbb{E}[\mathbf{x}\mathbf{x}^H]\mathbf{V})$ should be diagonal, which indicates $\mathbb{E}[\mathbf{x}\mathbf{x}^H] = \mathbf{V}\Sigma_{\mathbf{x}}\mathbf{V}^H$. Thus, the sufficiency of the condition is proved.

Similarly, we can apply Eq. (5.11) on both sides of Eq. (5.9), we can establish the necessity of the condition in Eq. (5.11). □

C.2 Proof of Theorem 5.2

Proof. Since $\mathbf{A} = \sum_{r=1}^{N} \lambda_r \cdot \underbrace{\mathbf{f}_r \circ \ldots \circ \mathbf{f}_r}_{\text{M times}}$, we have

$$a_{i_1 i_2 \cdots i_M} = \sum_{r=1}^{N} \lambda_r f_{r,i_1} f_{r,i_2} \cdots f_{r,i_M}, \tag{C.7}$$

where $f_{r,i}$ is the ith element of $\mathbf{f}_r$. Then, the tensor norm is

$$\begin{aligned}
||\mathbf{A}||_T^2 &= \sum_{i_1,i_2,\cdots,i_M} (\sum_{r=1}^{N} \lambda_r f_{r,i_1} f_{r,i_2} \cdots f_{r,i_M})^2 \\
&= \sum_{i_1,i_2,\cdots,i_M} (\sum_{r=1}^{N} \lambda_r f_{r,i_1} \cdots f_{r,i_M})(\sum_{t=1}^{N} \lambda_t f_{t,i_1} \cdots f_{t,i_M}) \\
&= \sum_{i_1,i_2,\cdots,i_M} \sum_{r,t} \lambda_r \lambda_t f_{r,i_1} \cdots f_{r,i_M} f_{t,i_1} \cdots f_{t,i_M} \\
&= \sum_{r,t} \lambda_r \lambda_t \sum_{i_1,i_2,\cdots,i_M=1}^{N} (f_{r,i_1} f_{t,i_1}) \cdots (f_{r,i_M} f_{t,i_M}) \\
&= \sum_{r,t} \lambda_r \lambda_t (\mathbf{f}_r^T \mathbf{f}_t)^M. \qquad \text{(C.8)}
\end{aligned}$$

Since $\mathbf{f}_r$ is orthonormal, $\mathbf{f}_r^T \mathbf{f}_t = 1$ holds if $r = t$; otherwise, $\mathbf{f}_r^T \mathbf{f}_t = 0$. Thus, we obtain $||\mathbf{A}||_T^2 = \sum_{r=1}^{N} \lambda_r^2$. □

Appendix D

Proofs for Chapter 6

D.1 Proof of Lemma 6.1

Proof. Let $\mathbf{V} = [\mathbf{f}_1, \cdots, \mathbf{f}_N]$ and $\Sigma = \mathrm{diag}([\lambda_1, \cdots, \lambda_N])$. Since $\mathbf{V}^{\mathrm{T}}\mathbf{V} = \mathbf{I}$, we have

$$\begin{aligned}
\mathbf{P}^k\mathbf{x} &= \underbrace{\mathbf{V}\Sigma\mathbf{V}^{\mathrm{T}}\mathbf{V}\Sigma\mathbf{V}^{\mathrm{T}}\cdots\mathbf{V}\Sigma\mathbf{V}^{\mathrm{T}}}_{k \quad \text{times}}\mathbf{x} \\
&= \mathbf{V}\Sigma^k\mathbf{V}^{\mathrm{T}}\mathbf{x} && \text{(D.1)} \\
&= \sum_{r=1}^{N}\lambda_r^k(\mathbf{f}_r^{\mathrm{T}}\mathbf{x})\mathbf{f}_r. && \text{(D.2)}
\end{aligned}$$

Since $\mathbf{H} = h(\mathbf{P}) = \sum_k \alpha_k \mathbf{P}^k$ is a polynomial graph filter, we can directly obtain

$$\begin{aligned}
\mathbf{H}\mathbf{x} &= \sum_k \sum_{r=1}^{N} \alpha_k \lambda_r^k(\mathbf{f}_r^{\mathrm{T}}\mathbf{x})\mathbf{f}_r && \text{(D.3)} \\
&= \sum_{r=1}^{N} h(\lambda_r)(\mathbf{f}_r^{\mathrm{T}}\mathbf{x})\mathbf{f}_r. && \text{(D.4)}
\end{aligned}$$

□

D.2 Proof of Theorem 6.1

Proof. Since the convolution filter $T_g(\mathbf{x})$ can be written in

$$T_g(\mathbf{x}) = \sum_{r=1}^{N} g(\lambda_r)\mathbf{f}_r(\mathbf{f}_r^{\mathrm{T}}\mathbf{x}), \tag{D.5}$$

the proof is straightforward by invoking with *Lemma* 6.1. □

References

[1] P. Sibani and H. J. Jensen, *Stochastic Dynamics of Complex Systems: From Glasses to Evolution*. London, UK: Imperial College Press, 2013.

[2] A. Ortega, P. Frossard, J. Kovaevi, J.M.F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," Proceedings of the IEEE, vol. 106, no. 5, pp. 808-828, Apr. 2018.

[3] Committee on Network Science for Future Army Applications, "Network science," National Research Council of the National Academies, Washington DC, Tech. Rep., 2005, the National Academies Press.

[4] T. G. Lewis, Network science: Theory and applications. John Wiley and Sons, 2011.

[5] A. L. Barabasi, Network Science. Cambridge University Press, 2016.

[6] M. O. Jackson, Social and economic networks. Princeton University Press, 2010.

[7] D. Easley and J. Kleinberg, Networks, crowds, and markets: Reasoning about a highly connected world. Cambridge University Press, 2010

[8] Z. Huang, C. Wang, M. Stojmenovic, and A. Nayak, "Characterization of cascading failures in interdependent cyberphysical systems," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2158-2168, Aug. 2015.

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

[10] M. I. Jordan, Learning in graphical models. Springer Science and Business Media, 1998, vol. 89.

[11] M. J. Wainwright, M. I. Jordan et al., "Graphical models, exponential families, and variational inference," *Foundations and Trends R in Machine Learning*, vol. 1, no. 1–2, pp. 1–305, 2008.

[12] V. P. Dwivedi, C. K. Joshi, T. Laurent, Y. Bengio, and X. Bresson, "Benchmarking graph neural networks," arXiv preprint arXiv:2003.00982, 2020.

[13] A. Sandryhaila, and J.M.F. Moura, "Discrete signal processing on graphs," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1644-1656, Apr. 2013.

[14] A. Clauset, C. R. Shalizi, and M. E. Newman, “Power-law distributions in empirical data,” *SIAM Review*, vol. 51, no. 4, pp. 661-703, Nov, 2009.

[15] R. Wagner, V. Delouille, and R. G. Baraniuk, “Distributed wavelet denoising for sensor networks,” in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, USA, Dec. 2006, pp. 373379.

[16] S.K. Narang, and A. Ortega, “Local two-channel critically sampled filter-banks on graphs,” in *Proc. of 17th IEEE International Conference on Image Processing (ICIP)*, Hong Kong, China, Sept. 2010, pp. 333-336.

[17] X. Zhu and M. Rabbat, “Approximating signals supported on graphs,” in *Proc. ICASSP*, Kyoto, Japan, Mar. 2012, pp. 3921-3924

[18] L. J. Grady and J. Polimeni, Discrete calculus: Applied analysis on graphs for computational science, USA: Springer Science and Business Media, 2010.

[19] C.G. Drake, S.J. Rozzo, H.F. Hirschfeld, N.P. Smarnworawong, E.D. Palmer, and B.L. Kotzin, “Analysis of the New Zealand Black contribution to lupus-like renal disease. Multiple genes that operate in a threshold manner,” *the Journal of Immunology*, vol. 154, no. 5, pp. 2441-2447, Mar. 1995

[20] S. Kok, and P. Domingos, “Learning Markov logic network structure via hypergraph lifting,” in *Proceedings of the 26th Annual International Conference on Machine Learning*, New York, USA, Jun. 2009, pp. 505-512.

[21] N. Meinshausen, and P. Bühlmann, “High-dimensional graphs and variable selection with the lasso,” *The Annals of Statistics*, vol. 34, no.3, 2006, pp. 1436-1462.

[22] S. M. Amin and B. F. Wollenberg, “Toward a smart grid: power delivery for the 21st century,” *IEEE Power and Energy Magazine*, vol. 3, no. 5, pp. 34-41, Sept. 2005.

[23] C. Berge, and E. Minieka, Graphs and Hypergraphs. Amsterdam, Nederland: North-Holland Pub. Co., 1973.

[24] X. Li, W. Hu, C. Shen, A. Dick, and Z. Zhang, “Context-aware hypergraph construction for robust spectral clustering,” *IEEE Transactions on Knowledge and Data Engineering*, vol.26, no. 10, pp. 2588-2597, July 2014.

[25] N. Yadati, M. Nimishakavi, P. Yadav, A. Louis, and P. Talukdar, “HyperGCN: Hypergraph Convolutional Networks for Semi-Supervised Classification.” arXiv:1809.02589, Sept. 2018.

[26] A. Mithani, G. M. Preston, and J. Hein. “Rahnuma: hypergraph-based tool for metabolic pathway prediction and network comparison,” *Bioinformatics*, vol. 25, no. 14, pp. 1831-1832, Jul. 2009.

[27] T. G. Kolda and B. W. Bader, “Tensor decompositions and applications,” *SIAM Review*, vol. 51, no. 3, pp. 455-500, Aug. 2009.

[28] S. Zhang, H. Zhang, H. Li and S. Cui, "Tensor-based spectral analysis of cascading failures over multilayer complex systems," in *Proceedings of 56th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, Monticello, USA, Oct. 2018, pp. 997-1004.

[29] K. C. Chang, K. Pearson, and T. Zhang, "On eigenvalue problems of real symmetric tensors," *Journal of Mathematical Analysis and Applications*, vol. 305, no. 1 pp. 416-422, Oct. 2008.

[30] P. Comon and B. Mourrain, "Decomposition of quantics in sums of powers of linear forms," *Signal Processing*, vol.53, no.2-3, pp. 93–107, Sept. 1996.

[31] P. Comon, G. Golub, L. H. Lim, and B. Mourrain, "Symmetric tensors and symmetric tensor rank," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 3, pp. 1254–1279, Sept. 2008.

[32] R. A. Horn, "The hadamard product," *Proc. Symp. Appl. Math*, Vol. 40, 1990.

[33] P. Symeonidis, A. Nanopoulos, and Y. Manolopoulos, "Tag recommendations based on tensor dimensionality reduction," in *Proceedings of the 2008 ACM Conference on Recommender Systems*, Lousanne, Switzerland, Oct. 2018, pp. 43-50.

[34] S. Liu, and G. Trenkler, "Hadamard, Khatri-Rao, Kronecker and other matrix products," *International Journal of Information and Systems Sciences*, vol. 4, no. 1, pp. 160-177, 2008.

[35] H. A. L. Kiers, "Towards a standardized notation and terminology in multiway analysis," *Journal of Chemometrics: A Journal of the Chemometrics Society*, vol. 14, no. 3, pp. 105-122, Jan. 2000.

[36] A. Afshar, J. C. Ho, B. Dilkina, I. Perros, E. B. Khalil, L. Xiong, and V. Sunderam, "Cp-ortho: an orthogonal tensor factorization framework for spatio-temporal data," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, Redondo Beach, CA, USA, Jan. 2017, p. 67.

[37] L. Qi, "Eigenvalues of a real supersymmetric tensor," *Journal of Symbolic Computation*, vol. 40, no. 6, pp. 1302-1324, Jun. 2005.

[38] L. Qi, "Eigenvalues and invariants of tensors," *Journal of Mathematical Analysis and Applications*, vol. 325, no. 2, pp. 1363-1377, Jan. 2007.

[39] L. H. Lim, "Singular values and eigenvalues of tensors: a variational approach," in *1st IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing*, Puerto Vallarta, Mexico, Dec. 2005, pp. 129-132.

[40] M. Ng, L. Qi, and G. Zhou, "Finding the largest eigenvalue of a nonnegative tensor," *SIAM Journal on Matrix Analysis and Applications*, vol.31, no.3, pp. 1090-1099, Feb. 2009.

[41] D. Cartwright, and B. Sturmfels, "The number of eigenvalues of a tensor," *Linear Algebra and its Applications*, vol. 438, no. 2, pp. 942-952, Jan. 2013.

[42] M. N. Marshall, "Sampling for qualitative research," *Family practice*, vol. 13, no. 6, pp. 522-526, Dec. 1996.

[43] S. Boccaletti, G. Bianconi, R. Criado, C. I. Del Genio, J. Gómez-Gardenes, M. Romance, I. Sendina-Nadal, Z. Wang, and M. Zanin, "The structure and dynamics of multilayer networks," *Physics Reports*, vol. 544, no. 1, pp. 1-122, Nov. 2014.

[44] S. V. Buldyrev, R. Parshani, G. Paul, H. E. Stanley, and S. Havlin, "Catastrophic cascade of failures in interdependent networks," *Nature*, vol. 464, no. 7291, pp. 1025-1028, Apr. 2010.

[45] Z. Zhang, W. An, and F. Shao, "Cascading failures on reliability in cyber-physical system," *IEEE Transactions on Reliability*, vol. 65, no. 4, pp. 1745-1754, Sept. 2016.

[46] H. Wang, Q. Li, G. DAgostino, S. Havlin, H. E. Stanley, and P. Van Mieghem, "Effect of the interconnected network structure on the epidemic threshold," *Physical Review E*, vol. 88, no. 2, p. 022801, Aug. 2013.

[47] C. Granell, S. Gómez, and A. Arenas, "Dynamical interplay between awareness and epidemic spreading in multiplex networks," *Physical Review Letters*, vol. 111, no. 12, p. 128701, Sept. 2013.

[48] M. Lelarge, "Diffusion and cascading behavior in random networks," *Games and Economic Behavior*, vol. 75, no. 2, pp. 752-775, July 2012.

[49] M. De Domenico, V. Nicosia, A. Arenas, and V. Latora, "Structural reducibility of multilayer networks," *Nature Communications*, vol. 6, no. 6864, pp. 1-9, Apr. 2015.

[50] M. De Domenico, A. Solé-Ribalta, E. Cozzo, M. Kivelä, Y. Moreno, M. A. Porter, S. Gómez, and A. Arenas, "Mathematical formulation of multilayer networks," *Physical Review X*, vol. 3, no. 4, p. 041022, Dec. 2013.

[51] M. J. Keeling and P. Rohani, *Modeling Infectious Diseases in Humans and Animals.* Princeton, NJ, USA: Princeton University Press, 2011.

[52] S. Gómez, A. Arenas, J. Borge-Holthoefer, S. Meloni, and Y. Moreno, "Discrete-time markov chain approach to contact-based disease spreading in complex networks," *EPL (Europhysics Letters)*, vol. 89, no. 3, p. 38009, Feb. 2010.

[53] Y. Wang, D. Chakrabarti, C. Wang, and C. Faloutsos, "Epidemic spreading in real networks: An eigenvalue viewpoint," in *Proc. of International Symposium on Reliable Distributed Systems*, Florence, Italy, Oct. 2003, pp. 25-34.

[54] E. Valdano, L. Ferreri, C. Poletto, and V. Colizza, "Analytical computation of the epidemic threshold on temporal networks," *Physical Review X*, vol. 5, no. 2, p. 021005, Apr. 2015.

[55] V. D. Blondel and Y. Nesterov, "Polynomial-time computation of the joint spectral radius for some sets of nonnegative matrices," *SIAM Journal on Matrix Analysis and Applications*, vol. 31, no. 3, pp. 865-876, Aug. 2009.

[56] Durrett, Richard, *Random Graph Dynamics*. Cambridge, MA, USA: Cambridge University Press, 2007.

[57] B. G. Horne, "Lower bounds for the spectral radius of a matrix," *Linear Algebra and its Applications*, vol. 263, pp. 261-273, Sept. 1997.

[58] H. Sayama, "Graph product multilayer networks: spectral properties and applications," *Journal of Complex Networks*, 2017. [Online]. Available: www.oxfordjournals.org. [Accessed Sept. 02, 2017].

[59] M. Newman, D. J. Watts, and S. H. Strogatz, "Random graph models of social networks," *Proceedings of the National Academy of Sciences*, vol. 99, no. 1, pp. 2566-2572, Feb. 2002.

[60] J. Shi, and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, pp. 888–905, Aug. 2000.

[61] A. Sandryhaila, and J. M. F. Moura, "Discrete signal processing on graphs: frequency analysis," *IEEE Transactions on Signal Processing*, vol. 62, no. 12, pp. 3042-3054, Apr. 2014.

[62] G. Ghoshal, V. Zlatić, G. Caldarelli, and M. E. J. Newman, "Random hypergraphs and their applications," *Physical Review E*, vol. 79, no. 6, p. 066118, Jun. 2009.

[63] K. Ahn, K. Lee, and C. Suh, "Hypergraph spectral clustering in the weighted stochastic block model," *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 5, pp. 959-974, May 2018.

[64] V. I. Voloshin, *Introduction to Graph and Hypergraph Theory*. New York, USA: Nova Science Publishers, 2009.

[65] T. Michoel, and B. Nachtergaele, "Alignment and integration of complex networks by hypergraph-based spectral clustering," *Physical Review E*, vol. 86, no. 5, p. 056111, Nov. 2012.

[66] B. Oselio, A. Kulesza, and A. Hero, "Information extraction from large multi-layer social networks," in *2015 IEEE International Conference on Acoustics, Speech and Signal Processing*, Brisbane, Australia, Apr. 2015, pp. 5451-5455.

[67] S. Barbarossa, and M. Tsitsvero, "An introduction to hypergraph signal processing," in *Proc of Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, Mar. 2016, pp. 6425-6429.

[68] K. Fukunaga, S. Yamada, H. S. Stone, and T. Kasai, "A representation of hypergraphs in the euclidean space," *IEEE Transactions on Computers*, vol. 33, no. 4, pp. 364-367, Apr. 1984.

[69] A. Banerjee, C. Arnab, and M. Bibhash, "Spectra of general hypergraphs," *Linear Algebra and its Applications*, vol. 518, pp. 14-30, Dec. 2016.

[70] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic, "Signal denoising on graphs via graph filtering," in *Proc. of Signal and Information Processing (GlobalSIP)*, Atlanta, GA, USA, Dec. 2014, pp. 872-876.

[71] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovačević, "Adaptive graph filtering: multiresolution classification on graphs," in *2013 IEEE Global Conference on Signal and Information Processing*, Austin, TX, USA, Feb. 2014, pp. 427-430.

[72] A. Sandryhaila, and J. M. F. Moura, "Discrete signal processing on graphs: graph Fourier transform," in *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, BC, Canada, May 2013, pp. 26-31.

[73] A. Sandryhaila, and J. M. F. Moura, "Discrete signal processing on graphs: graph filters," in *Proceedings of 2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, Vancouver, Canada, May 2013, pp. 6163-6166.

[74] G. Li, L. Qi, and G. Yu, "The Z-eigenvalues of a symmetric tensor and its application to spectral hypergraph theory," *Numerical Linear Algebra with Applications*, vol. 20, no. 6, pp. 1001-1029, Mar. 2013.

[75] K. J. Pearson, and T. Zhang, "On spectral hypergraph theory of the adjacency tensor," *Graphs and Combinatorics*, vol. 30, no. 5, pp. 1233-1248, Sept. 2014.

[76] M. A. Henning, and Y. Anders, "2-Colorings in k-regular k-uniform hypergraphs," *European Journal of Combinatorics*, vol. 34, no. 7, pp. 1192-1202, Oct. 2013.

[77] E. Robeva, "Orthogonal decomposition of symmetric tensors," *SIAM Journal on Matrix Analysis and Applications*, vol. 37, no. 1, pp. 86-102, Jan. 2016.

[78] A. Cichocki, D. Mandic, L. Lathauwer, G. Zhou, Q. Zhao, C. Caiafa, and H. A. Phan, "Tensor decompositions for signal processing applications: from two-way to multiway component analysis," *IEEE Signal Processing Magazine*, vol. 32, no. 2, pp. 145-163, Feb. 2015.

[79] D. Silva, and A. Pereira, 'Tensor techniques for signal processing: algorithms for Canonical Polyadic decomposition,' University Grenoble Alpes, 2016.

[80] V. Nguyen, K. Abed-Meraim, and N. Linh-Trung, "Fast tensor decompositions for big data processing," in *Proceedings of 2016 International Conference on Advanced Technologies for Communications*, Hanoi, Vietnam, Oct, 2016, pp. 215-221.

[81] E. O. Brigham and E. O. Brigham, *The Fast Fourier Transform and its Applications*, New Jersey: Prentice Hall, 1988.

[82] S. Mallat, *A Wavelet Tour of Signal Processing, 3rd ed.* New York, NY, USA: Academic, 2008.

[83] V. Rödl, and J. Skokan, "Regularity lemma for k-uniform hypergraphs," *Random Structures and Algorithms*, vol. 25, no. 1, pp. 1-42, Jun. 2004.

[84] Z. K. Zhang, and C. Liu, "A hypergraph model of social tagging networks," *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2010, no.10 , P10005, Oct. 2010.

[85] K. Pearson, and T. Zhang, "Eigenvalues on the adjacency tensor of products of hypergraphs," *Int. J. Contemp. Math. Sci*, vol. 8, pp. 151-158, Jan. 2013.

[86] J. M. Mendel, "Use of higher-order statistics in signal processing and system theory: an update," *Advanced Algorithms and Architectures for Signal Processing*, vol. 975, pp. 126-145, Feb. 1988.

[87] J. M. Mendel, "Tutorial on higher-order statistics (spectra) in signal processing and system theory: theoretical results and some applications," *Proceedings of the IEEE*, vol. 79, no. 3, pp. 278-305, Mar. 1991.

[88] A. K. Nandi, *Blind estimation using higher-order statistics.* USA: Springer, 2013.

[89] M. S. Choudhury, S. L. Shah, and N. F. Thornhill, "Diagnosis of poor control-loop performance using higher-order statistics," *Automatica*, vol. 40, no. 10, pp. 1719-1728, Oct. 2004.

[90] O. A. Dobre, Y. Bar-Ness, and W. Su, "Higher-order cyclic cumulants for high order modulation classification," in *Proceedings of IEEE Military Communications Conference*, Boston, MA, USA, Oct. 2003, pp. 112-117.

[91] M. B. Priestley, *Spectral Analysis and Time Series.* London, UK: Academic Press, 1981.

[92] A. N. A. N. T. H. R. A. M. Swami, and J. M. Mendel, "Time and lag recursive computation of cumulants from a state-space model," *IEEE Transactions on Automatic Control*, vol. 35, no. 1, pp. 4-17, Jan. 1990.

[93] M. Hein, S. Setzer, L. Jost, and S. S. Rangapuram, "The total variation on hypergraphs-learning on hypergraphs revisited," in *Advances in Neural Information Processing Systems*, Lake Tahoe, Nevada, Dec. 2013, pp. 2427-2435.

[94] G. Chen, J. Zhang, F. Wang, C. Zhang, and Y. Gao, "Efficient multi-label classification with hypergraph regularization," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Miami, FL, USA, Jun. 2009, pp. 1658-1665.

[95] S. Chen, A. Sandryhaila, and J. Kovačević, "Sampling theory for graph signals," in *Proc. of Acoustics, Speech and Signal Processing (ICASSP)*, Brisbane, Australia, Apr. 2015, pp. 3392-3396.

[96] G. E. Batley, and D. Gardner, "Sampling and storage of natural waters for trace metal analysis," *Water Research*, vol. 11, no. 9, pp. 745-756, 1977.

[97] S. Chen, R. Varma, A. Sandryhaila, and J. Kovačević, “Discrete signal processing on graphs: sampling theory,” *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6510-6523, Dec. 2015.

[98] V. I. Paulsen, and R. R. Smith, “Multilinear maps and tensor norms on operator systems,” *Journal of Functional Analysis*, vol. 73, no. 2, pp. 258-276, Aug. 1987.

[99] S. Chen, F. Cerda, P. Rizzo, J. Bielak, J. H. Garrett, and J. Kovačević, “Semi-supervised multiresolution classification using adaptive graph filtering with application to indirect bridge structural health monitoring,” *IEEE Transactions on Signal Processing*, vol. 62, no. 11, pp. 2879-2893, Mar. 2014.

[100] S. McCanne, and M. J. Demmer, “Content-based segmentation scheme for data compression in storage and transmission including hierarchical segment representation,” U.S. Patent, 6667700, Dec., 2003.

[101] K. Sayood, *Introduction to Data Compression*, Burlington, Massachusetts, USA: Morgan Kaufmann, 2017.

[102] D. Salomon, *Data Compression: the Complete Reference*. London, UK: Springer, 2004.

[103] A. Bretto, and L. Gillibert, “Hypergraph-based image representation,” in *International Workshop on Graph-Based Representations in Pattern Recognition*, Berlin, Heidelberg, Apr. 2005, pp. 1-11.

[104] A. Morar, F. Moldoveanu, and E. Gröller, “Image segmentation based on active contours without edges,” in *Proc. of 2012 IEEE 8th International Conference on Intelligent Computer Communication and Processing*, Cluj-Napoca, Romania, Sept. 2012, pp. 213-220.

[105] F. Maxwell Harper and Joseph A. Konstan, “The movieLens datasets: history and context,” *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 5, no. 19, Jan. 2016. [Online] DOI=http://dx.doi.org/10.1145/2827872

[106] D. Zhou, J. Huang, and B. Schölkopf, “Learning with hypergraphs: clustering, classification, and embedding,” *Advances in Neural Information Processing Systems*. pp. 1601-1608, Sept. 2007.

[107] U. V. Luxburg, “A tutorial on spectral clustering,” *Statistics and Computing*, vol. 17, no.4, pp. 395-416, Dec. 2007.

[108] D. Dua, and E. K. Taniskidou, UCI Machine Learning Repository, 2017. [Online]. Avaiable: http://archive.ics.uci.edu/ml. [Irvine, CA: University of California, School of Information and Computer Science].

[109] P. J. Rousseeuw, “Silhouettes: a graphical aid to the interpretation and validation of cluster analysis,” *Journal of Computational and Applied Mathematics*, vol. 20, pp. 53-65, Nov. 1987.

[110] D. L. Medin, and M. Schaffer, "Context theory of classification learning," *Psychological Review*, vol. 85, no. 3, pp. 207-238, 1978.

[111] X. Zhu, and Z. Ghahramani, "Learning from labeled and unlabeled data with label propagation," *CMU CALD Tech Report*, vol. 02, no. 107, 2002.

[112] J. Jung, S. Chun, and K. Lee, "Hypergraph-based overlay network model for the Internet of Things," *2015 IEEE 2nd World Forum on Internet of Things (WF-IoT)*, Milan, Italy, Dec. 2015, pp. 104-109.

[113] V. Zlatić, G. Ghoshal, and G. Caldarelli, "Hypergraph topological quantities for tagged social networks," *Physical Review E*, vol. 80, no. 3, p. 036118, Sept. 2009.

[114] I. Konstas, and M. Lapata, "Unsupervised concept-to-text generation with hypergraphs," in *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Montreal, Canada, Jun. 2012, pp. 752-761.

[115] H. Boley, "Directed recursive labelnode hypergraphs: a new representation-language," *Artificial Intelligence*, vol. 9, no. 1, pp. 49-85, Aug. 1977.

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

[117] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83-98, Apr. 2013.

[118] B. Xu, H. Shen, Q. Cao, Y. Qiu, and X. Cheng, "Graph wavelet neural network," arXiv:1904.07785.

[119] T. N. Kipf, and M. Welling, "Semi-supervised classification with graph convolutional networks," arXiv:1609.02907.

[120] J. Yu, D. Tao, and M. Wang, "Adaptive hypergraph learning and its application in image classification," *IEEE Transactions on Image Processing*, vol. 21, no. 7, pp. 3262-3272, Mar. 2012.

[121] Y. Huang, Q. Liu, S. Zhang, and D. N. Metaxas, "Image retrieval via probabilistic hypergraph ranking," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Francisco, CA, Jun. 2010, pp. 3376-3383.

[122] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," arXiv:1901.08150.

[123] S. Zhang, Z. Ding, and S. Cui, "Introducing Hypergraph Signal Processing: Theoretical Foundation and Practical Applications," *IEEE Internet of Things Journal*, vol. 7, pp. 639 - 660, Jan. 2020.

[124] S. Zhang, S. Cui, and Z. Ding, “Hypergraph spectral analysis and processing in 3d point cloud,” arXiv:2001.02384.

[125] S. Zhang, S. Cui, and Z. Ding, “Point cloud segmentation based on hypergraph spectral clustering,” arXiv:2001.07797.

[126] J. Shi, and J. M. Moura, “Graph signal processing: modulation, convolution, and sampling,” arXiv:1912.06762.

[127] D. I. Shuman, B. Ricaud, and P. Vandergheynst, “A windowed graph Fourier transform,” in *2012 IEEE Statistical Signal Processing Workshop*, Ann Arbor, MI, USA, Aug. 2012, pp. 133-136.

[128] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, “Stationary graph processes and spectral estimation,” *IEEE Transactions on Signal Processing*, vol. 65, no. 22, pp. 5911-5926, Aug. 2017.

[129] 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, Aug. 2017.

[130] K. I. Park, *Fundamentals of probability and stochastic processes with applications to communications*. Springer International Publishing, 2018.

[131] A. K. Cherri, and M. A. Karim, “Optical symbolic substitution: edge detection using Prewitt, Sobel, and Roberts operators,” *Applied Optics*, vol. 28, no. 21, pp. 4644-4648, Nov. 1989.

[132] R. Lionnie, I. K. Timotius, and I. Setyawan, “Performance comparison of several preprocessing methods in a hand gesture recognition system based on nearest neighbor for different background conditions,” *Journal of ICT Research and Applications*, vol. 6, no. 3, pp. 183-194, 2012.

[133] A. Aldoma, Z. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, “Tutorial: point cloud library: three-dimensional object recognition and 6 dof pose estimation,” *IEEE Robotics and Automation Magazine*, vol. 19, no. 3, pp. 80-91, Sep. 2012.

[134] R. B. Rusu and S. Cousins, “3D is here: point cloud library (pcl),” in *2011 IEEE International Conference onRobotics and Automation*, Shanghai, China, May 2011, pp. 1–4.

[135] S. Chen, D. Tian, C. Feng, A. Vetro and J. Kovačević, “Fast resampling of three-dimensional point clouds via graphs,” *IEEE Transactions on Signal Processing*, vol. 66, no. 3, pp. 666-681, Feb, 2018.

[136] J. Ryde, and H. Hu, “3D mapping with multi-resolution occupied voxel lists” *Autonomous Robots*, vol. 28, no. 2, pp. 169-185. Apr. 2010.

[137] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard, "Octomap: an efficient probabilistic 3D mapping framework based on octrees," *Autonomous Robots*, pp. 189–206, Apr. 2013.

[138] J. Peng and C. C. Jay Kuo, "Geometry-guided progressive lossless 3D mesh coding with octree (OT) decomposition," *ACM Trans. Graph. Proceedings of ACM SIGGRAPH*, vol. 24, no. 3, pp. 609–616, Jul. 2005.

[139] Y. Schoenenberger, J. Paratte, and P. Vandergheynst, "Graph-based denoising for time-varying point clouds," in *2015 3DTV-Conference: The True Vision-Capture, Transmission and Display of 3D Video (3DTV-CON)*, Lisbon, Portugal, Aug. 2015, pp. 1-4.

[140] J. Qi, W. Hu and Z. Guo, "Feature preserving and uniformity-controllable point cloud simplification on graph," *2019 IEEE International Conference on Multimedia and Expo (ICME)*, Shanghai, China, 2019, pp. 284-289.

[141] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3D point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927-941, Nov. 2008.

[142] A. Nurunnabi, D. Belton and G. West, "Robust segmentation in laser scanning 3D point cloud data," *2012 International Conference on Digital Image Computing Techniques and Applications (DICTA)*, Fremantle, WA, 2012, pp. 1-8.

[143] H. Hoppe, T. De Rose, and T. Duchamp, "Surface reconstruction from unorganized points," in *Proceedings of ACM SIGGRAPH*, vol. 26, no. 2, pp. 71–78, 1992.

[144] M. Pauly, R. Keiser, and M. Gross, "Multi-scale feature extraction on point-sampled surfaces," *Eurographics*, vol. 22, no. 3, 2003.

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

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

[147] M. Grant and S. Boyd. "Graph implementations for nonsmooth convex programs," *Recent Advances in Learning and Control (a tribute to M. Vidyasagar).*

c28-e30

[148] V. Blondel, S. Boyd, and H. Kimura, editors, pages 95-110, *Lecture Notes in Control and Information Sciences*, Springer, 2008. http://stanford.edu/ boyd/graph_dcp.html.

[149] Y. Sun, S. Schaefer, and W. Wang, "Denoising point sets via l0 minimization," *Computer Aided Geometric Design*, vol. 35, pp. 2–15, May 2015.

[150] J. Zeng, G. Cheung, M. Ng, J. Pang, and Yang, C. "3d point cloud denoising using graph laplacian regularization of a low dimensional manifold model," 2018, arXiv:1803.07252.

[151] C. Dinesh, G. Cheung, I. V. Bajic, and C. Yang, "Fast 3d point cloud denoising via bipartite graph approximation and total variation," 2018, arXiv: 1804.10831.

[152] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Numerical geometry of non-rigid shapes", Springer, 2008. ISBN: 978-0-387-73300-5.

[153] A. M. Bronstein, M. M. Bronstein, and R. Kimmel, "Efficient computation of isometry-invariant distances between surfaces", *SIAM J. Scientific Computing*, vol. 28, no. 5, pp. 1812-1836, 2006.

[154] A. M. Bronstein *et al.*, "SHREC 2010: robust large-scale shape retrieval benchmark", *Proc. EUROGRAPHICS Workshop on 3D Object Retrieval (3DOR)*, May 2010, pp. 71–78..

[155] A. M. Bronstein *et al.*, "SHREC 2010: robust feature detection and description benchmark", *Proc. EUROGRAPHICS Workshop on 3D Object Retrieval (3DOR)*, vol. 2, no. 5, May 2010, p. 6.

[156] L.Yi *et al.*, "Large-scale 3d shape reconstruction and segmentation from shapenet core55," 2017, arXiv: 1710.06104.

[157] A. X. Chang *et al.*, "Shapenet: an information-rich 3D model repository," 2015, arXiv: 1512.03012.

[158] J. Pang, and G. Cheung, "Graph Laplacian regularization for image denoising: analysis in the continuous domain," *IEEE Transactions on Image Processing*, vol. 26, no. 4, pp. 1770-1785, Apr. 2017.

[159] S. W. Cheng, and M. K. Lau, "Denoising a point cloud for surface reconstruction," 2017, arXiv: 1704.04038.

[160] W. Hu, X. Gao, G. Cheung, and Z. Guo, "Feature graph learning for 3D point cloud denoising," *IEEE Transactions on Signal Processing*, vol.68, pp. 2841-2856, Mar. 2020.

[161] A. Nguyen, and L. Bac, "3D point cloud segmentation: a survey," in *6th IEEE Conference on Robotics, Automation and Mechatronics*, Manila, Philippines, Mar. 2013, pp. 225-230.

[162] J. Cheng, M. Qiao, W. Bian, and D. Tao, "3D human posture segmentation by spectral clustering with surface normal constraint", *Signal Processing*, vol. 91, no. 9, pp. 2204-2212, Sep. 2011.

[163] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International Conference on Machine learning (ICML-03)*, Washington DC, USA, Aug. 2003, pp. 912–919.

[164] S. White and P. Smyth, "A spectral clustering approach to finding communities in graphs," in *Proceedings of the 2005 SIAM International Conference on Data Mining*, Newport Beach, CA, USA, 2005, pp. 274– 285.

[165] T. Buhler and M. Hein, "Spectral clustering based on the graph p-laplacian," in *Proceedings of the 26th Annual International Conference on Machine Learning*, New York, NY, USA, Jun. 2009, pp. 81–88.

[166] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Francisco, CA, USA, 2016, pp. 855–864.

[167] M. Niepert, M. Ahmed, and K. Kutzkov, "Learning convolutional neural networks for graphs," in *International Conference on Machine Learning*, New York, NY, USA, Jun. 2016, pp. 2014–2023.

[168] S. Zhang, S. Cui, and Z. Ding, "Hypergraph-based image processing," in *2020 IEEE International Conference on Image Processing (ICIP)*, Abu Dhabi, United Arab Emirates, 2020, pp. 216–220.

[169] J. Klicpera, A. Bojchevski, and S. Gunnemann, "Predict then propagate: Graph neural networks meet personalized pagerank," arXiv preprint arXiv:1810.05997, 2018.

[170] S. Abu-El-Haija, A. Kapoor, B. Perozzi, and J. Lee, "N-gcn: Multiscale graph convolution for semi-supervised node classification," arXiv preprint arXiv:1802.08888, 2018.

[171] S. Barbarossa and S. Sardellitti, "Topological signal processing over simplicial complexes," *IEEE Transactions on Signal Processing*, Mar. 2020.

[172] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web." Stanford InfoLab, Tech. Rep., 1999.

[173] X. Bresson and T. Laurent, "Residual gated graph convnets," arXiv preprint arXiv:1711.07553, 2017.

[174] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Advances in Neural Information Processing Systems*, Long Beach, USA, Dec. 2017, pp. 1024–1034.

[175] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein, "Geometric deep learning on graphs and manifolds using mixture model cnns," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Hawaii, USA, Jul. 2017, pp. 5115–5124.

[176] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio, "Graph attention networks," arXiv preprint arXiv:1710.10903, 2017.

[177] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec, "Hierarchical graph representation learning with differentiable pooling," in *Advances in Neural Information Processing Systems*, Montreal, Canada, Dec. 2018, pp. 4800–4810.

[178] H. Pei, B. Wei, K. C. C. Chang, Y. Lei, and B. Yang, "Geom-gcn: Geometric graph convolutional networks," arXiv preprint arXiv:2002.05287, 2020.

[179] S. Abu-El-Haija, B. Perozzi, A. Kapoor, N. Alipourfard, K. Lerman, H. Harutyunyan, G. V. Steeg, and A. Galstyan, "Mixhop: Higher-order graph convolutional architectures via sparsified neighborhood mixing," arXiv preprint arXiv:1905.00067, 2019.

[180] J. Atwood and D. Towsley, "Diffusion-convolutional neural networks," in *Advances in Neural Information Processing Systems*, 2016, pp. 1993– 2001.

[181] K. Xu, W. Hu, J. Leskovec, and S. Jegelka, "How powerful are graph neural networks?" arXiv preprint arXiv:1810.00826, 2018.

[182] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[183] J. Pons, J. Miralles, and J. M. Ibanez, "Legendre expansion of the kernel: Influence of high order terms," *Astronomy and Astrophysics Supplement Series*, vol. 129, no. 2, pp. 343–351, 1998.

[184] S. Linnainmaa, "Taylor expansion of the accumulated rounding error," *BIT Numerical Mathematics*, vol. 16, no. 2, pp. 146–160, 1976.

[185] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore, "Automating the construction of internet portals with machine learning," *Information Retrieval*, vol. 3, no. 2, pp. 127–163, Jul. 2000.

[186] A. Bojchevski and S. Gunnemann, "Deep gaussian embedding of graphs: Unsupervised inductive learning via ranking," arXiv preprint arXiv:1707.03815, 2017.

[187] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Galligher, and T. EliassiRad, "Collective classification in network data," *AI magazine*, vol. 29, no. 3, pp. 93–93, 2008.

[188] G. Namata, B. London, L. Getoor, B. Huang, and U. EDU, "Querydriven active surveying for collective classification," in *10th International Workshop on Mining and Learning with Graphs*, vol. 8, Edinburgh, Scotland, Jul. 2012.

[189] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[190] J. Lv, X. Chen, J. Huang, and H. Bao, "Semi-supervised mesh segmentation and labeling," in *Computer Graphics Forum*, vol. 31, no. 7. Wiley Online Library, 2012, pp. 2241–2248.

[191] S. Zhang, S. Cui, and Z. Ding, "Hypergraph spectral clustering for point cloud segmentation," *IEEE Signal Processing Letters*, vol. 27, pp. 1655– 1659, 2020.

[192] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 652–660.

[193] S. Yan, Y. Xiong, and D. Lin, "Spatial temporal graph convolutional networks for skeleton-based action recognition," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, Apr. 2018.

[194] T. N. Kipf, and M. Welling. "Variational graph auto-encoders.," arXiv preprint arXiv:1611.07308, 2016.

[195] H. Gao, amd S. Ji, "Graph u-nets," in *International Conference on Machine Learning*, pp. 2083-2092, May 2019.

[196] Y. Yu, K. Chan, C. You, C. Song, and Y. Ma, "Learning diverse and discriminative representations via the principle of maximal coding rate reduction," arXiv preprint arXiv:2006.08558, 2020.