COPPE
Instituto Alberto Luiz Coimbra de
Pós-Graduação e Pesquisa de Engenharia
UFRJ

DATA-DRIVEN PROCESSING OF GRAPH SIGNALS FOR ANOMALY DETECTION AND FORECASTING

Gabriela Lewenfus

Dissertação de Mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Wallace Alves Martins

Rio de Janeiro
Setembro de 2020

DATA-DRIVEN PROCESSING OF GRAPH SIGNALS FOR ANOMALY DETECTION AND FORECASTING

Gabriela Lewenfus

DISSERTAÇÃO SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA DA UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE MESTRE EM CIÊNCIAS EM ENGENHARIA ELÉTRICA.

Orientador: Wallace Alves Martins

Aprovada por: Prof. Wallace Alves Martins
Prof. Markus Vinícius Santos Lima
Prof. César Javier Niche Mazzeo
Prof. Juliano Bandeira Lima

RIO DE JANEIRO, RJ – BRASIL
SETEMBRO DE 2020

Lewenfus, Gabriela

Data-driven processing of graph signals for anomaly detection and forecasting /Gabriela Lewenfus. – Rio de Janeiro: UFRJ/COPPE, 2020.

XX, 140 p.: il.; 29, 7cm.

Orientador: Wallace Alves Martins

Dissertação (mestrado) – UFRJ/COPPE/Programa de Engenharia Elétrica, 2020.

Referências Bibliográficas: p. 111 – 128.

1. graph signal processing. 2. vertex-frequency analysis. 3. graph neural networks. I. Martins, Wallace Alves. II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia Elétrica. III. Título.

A Alzira e Cyna.

Agradecimentos

Antes de tudo, gostaria de agradecer imensamente aos meus pais por todo apoio que me deram em minha vida, por terem me permitido e me incentivado a estudar por mais tempo. Palavras não são suficientes para agradecer a tudo que eles fizeram e fazem por mim e sem eles nada teria sido possível. Agradeço a minha irmã e melhor amiga Suzana por ser sempre meu porto seguro com quem posso contar. Agradeço também ao meu cunhado Felipe, que considero irmão, por estar sempre disposto a me ajudar.

Como meu curso de graduação não tinha trabalho de conclusão de curso, quero aproveitar este espaço para agradecer a todos que fizeram parte da minha trajetória acadêmica desde 2014.

Agradeço à minha amiga de infância Jullyana por toda amizade e por ter tido sua companhia em boa parte da graduação, inclusive na nossa colação, depois de anos em escolas diferentes foi um grande presente pra mim.

Quero agradecer aos meus amigos e professores do curso de Biomedicina. Em especial, agradeço a Ana Paula, Desiree, Luciana, Maria e Yuri que estiveram comigo na maior parte do tempo. Agradeço ao Andrei por todos os ensinamentos sobre ciência, pela paciência e por todos os momentos divertidos no laboratório. Por fim, agradeço ao Kleber pela parceria, carinho e amizade.

Agradeço também a todos os professores e amigos do curso de matemática aplicada. Gostaria de agradecer em especial a alguns professores que marcaram esse período. Agradeço ao Felipe Acker pelas aulas “desafiadoras"que me ensinaram a estudar cada vez melhor e gostar mais ainda de matemática. Agradeço ao Bernardo e ao César por serem não apenas professores mas também grandes amigos. Não poderia deixar de citar alguns amigos com quem dividi boa parte dos meus dias: Alexandre, Bruno, Cláudio, Cynthia, Felipe, Gabriella, Gabriel, Hugo, Iago, Lucas Porto, Lucas Xavier, Pedro Gil, Pedro Paixão, Pedro Schmidt, Pedro, Ricardo, Rodrigo e Vitor. Obrigada em especial ao Patrick, uma pessoa fundamental na minha jornada pela matemática, pela amizade, estudos, piadas e almoços. Por fim, agradeço ao Diego por ter sido maravilhoso pra mim, ter me ajudado a descobrir e entender meus sonhos profissionais e pelo apoio em absolutamente tudo.

Agradeço ao Gabriel, Igor, Lucas, Markus, Matheus, Rafael, Thadeu, Wesley e Vinícius pelos ensinamentos, momentos divertidos e acima de tudo amizade que pretendo levar pra vida. Eles são pessoas incríveis, admiro muito cada um e agradeço imensamente por terem me acolhido nessa reta final do mestrado. Agradeço ao professor Marcello Campos pela oportunidade de participar de um projeto onde pude conhecer e trabalhar com boa parte das pessoas citadas anteriormente, e ter a experiência de trabalhar com um problema real. Agradeço à dona Edinaval por alegrar meus dias com sua enorme simpatia. Quero fazer também um agradecimento mais que especial à Domenica e à Rebeca que também me acompanharam nessa reta final. A amizade delas foi sem dúvida o melhor presente de 2019.

Agradeço ao meu orientador Wallace por tudo. Obrigada por me ensinar ciência, pela orientação na escrita, pelas oportunidades e por fazer tudo isso sempre com muita paciência. Se eu pudesse voltar no tempo, não escolheria outra pessoa para isso.

Gostaria de aproveitar também para agradecer ao João Guedes da Franca, meu primeiro orientador. Não tenho nem palavras para descrever o quão importante ele foi na minha trajetória acadêmica, pois com ele dei meus primeiros passos na ciência. Infelizmente, ele não pôde me ver alcançando essa meta, porém guardarei para sempre, com muito carinho, tudo que ele me ensinou.

Resumo da Dissertação apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Mestre em Ciências (M.Sc.)

PROCESSAMENTO DE SINAIS EM GRAFOS APLICADO À DETECÇÃO DE ANOMALIAS E PREDIÇÃO DE SÉRIES TEMPORAIS

Gabriela Lewenfus

Setembro/2020

Orientador: Wallace Alves Martins

Programa: Engenharia Elétrica

Processamento de sinais em grafos (GSP) é uma nova área que busca estender a teoria e as técnicas clássicas de processamento de sinais para analisar e processar dados definidos sobre grafos. Nesta dissertação, apresentamos uma revisão sobre tópicos fundamentais de GSP tais como análise de Fourier, amostragem e análise vértice-frequência (VFA), e propomos duas aplicações distintas de GSP. Na primeira, aplicamos VFA no problema de detecção de anomalias em sinais em grafos (GSs) variantes no tempo. No caso particular de localizar uma estação climática defeituosa, a acurácia obtida na detecção de pequenas variações de temperatura por algoritmos de detecção de *outliers* aumenta quando VFA é utilizado para extração de atributos. A segunda aplicação proposta nesta dissertação combina GSP e redes neurais recorrentes para prever e interpolar GSs simultaneamente. O modelo proposto, *spectral graph gated recurrent unit*, superou métodos do estado-da-arte, especialmente quando se tem acesso apenas a uma pequena fração do sinal de interesse, considerando dois conjuntos de dados distintos: temperatura nos Estados Unidos e velocidade do tráfego em Seattle.

Abstract of Dissertation presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Master of Science (M.Sc.)

DATA-DRIVEN PROCESSING OF GRAPH SIGNALS FOR ANOMALY DETECTION AND FORECASTING

Gabriela Lewenfus

September/2020

Advisor: Wallace Alves Martins

Department: Electrical Engineering

Graph signal processing (GSP) is an emerging field that extends traditional signal processing theory and techniques to analyze and process data defined over graphs. This dissertation presents fundamental topics of GSP, such as Fourier analysis, sampling graph signals, and vertex-frequency analysis (VFA), and proposes two different applications. In the first one, we apply VFA to the problem of anomaly detection in time-varying graph signals. In the particular example of localizing a malfunctioning weather station, the accuracy achieved by outlier detection algorithms is improved when fed with VFA-extracted features to detect small drifts in temperature measurements. The second GSP application proposed in this dissertation combines GSP and recurrent neural networks in order to jointly forecast and interpolate graph signals. The proposed learning model, namely *spectral graph gated recurrent unit*, outperforms state-of-the-art deep learning techniques, especially when only a small fraction of the graph signals is accessible, considering two distinct real world datasets: temperatures in the US and speed flow in Seattle.

Sumário

Lista de Figuras

Lista de Tabelas

List of Abbreviations

ASP algebraic signal processing.

CNN convolutional neural network.

CSF cumulative spectral function.

CWT continuous wavelet transform.

DCT discrete cosine transform.

DFT discrete Fourier transform.

DL deep learning.

DSP discrete signal processing.

EEG eletroencephalography.

FC fully connected.

fMRI functional resonance magnetic imaging.

GCN graph convolutional neural network.

GFT graph Fourier transform.

GPC Gaussian process classifier.

GRU gated recurrent unit.

GS graph signal.

GSP graph signal processing.

IF isolation forest.

LOF local outlier factor.

LSTM long short term memory.

MAE mean absolute error.

MAPE mean absolute percentate error.

ML machine learning.

MSE mean squared error.

NN neural network.

OCSVM one class support vector machine.

RMSE root mean squared error.

RNN recurrent neural network.

SG-GRU spectral graph gated recurrent unit.

SGC spectral graph convolution.

SGWT spectral graph wavelet transform.

SSL semi-supervised learning.

ST spatiotemporal.

STFT short time Fourier transform.

STGCN spatiotemporal graph graph convolution network.

TGC traffic graph convolution.

VFA vertex-frequency analysis.

WGFT windowed graph Fourier transform.

WSN wireless sensor networks.

List of Symbols

A_{nm} n^{th} row and m^{th} column of matrix $\mathbf{A}$.

$\mathbf{A}$ adjacency matrix.

$\mathbf{X}^t$ The vectors $\mathbf{X}^t = [\mathbf{x}^{t-\tau}\, \mathbf{x}^{t-\tau+1} \ldots \mathbf{x}^{t-1}]^{\mathrm{T}}$.

$\mathbf{x}$ vector representation of graph signal.

$\mathbf{x}^t$ snapshot t of graph signal $\mathbf{x}$.

$\mathcal{G}$ graph representation.

x_n n^{th} entry of vector $\mathbf{x}$.

x_t timestep t of the discrete signal $\mathbf{x}$.

$*$ convolution operator.

$\hat{\mathbf{x}}_{\mathcal{F}}$ the frequency content of the GS $\mathbf{x}$ restricted to the set $\mathcal{F}$.

$\mathrm{BL}_{\xi}(\mathbf{U})$ the space of bandlimited graph signals with bandwidth ξ.

$\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$ the space of bandlimited graph signals with support $\mathcal{F}$.

$|\mathcal{A}|$ cardinality of set $\mathcal{A}$.

$\overline{\mathcal{S}}$ the complement of the set $\mathcal{S}$ with respect to $\mathcal{V}$.

$\mathbf{1}$ constant vector.

$\boldsymbol{\delta}_n$ vector of zeros except for the n^{th} entry, which is 1.

$\frac{\partial}{\partial x}\mathbf{y}$ partial derivative of $\mathbf{y}$ with respect to x.

$\mathcal{E}$ set of graph edges.

λ_k k^{th} Laplacian/adjacency eigenvalue.

U_{nk} n^{th} row and k^{th} column of matrix $\mathbf{U}$.

$\mathbf{U}$ matrix of Laplacian eigenvectors.

$\|\cdot\|_2$ Euclidean norm.

$\mathbb{E}[X]$ expected value of X.

$\mathcal{F}$ subset of graph frequencies.

∇ gradient operator.

$\mathcal{N}_h(m)$ h-hop neighborhood around node n.

$d_{\mathcal{G}}(\cdot)$ hop-distance.

$\boldsymbol{\mathcal{I}}$ incidence matrix.

$\mathbb{1}$ indicator function.

$\langle\,,\,\rangle$ inner product operator.

$\mathbb{Z}$ set of integer numbers.

$\mathbf{\Phi}_{\mathcal{S}}$ interpolation operator $\mathcal{S} \rightarrow \mathcal{V}$.

$\mathbf{L}$ Laplacian matrix.

R Number of wavelet scales.

$\mathcal{L}$ loss function.

$m_k(\cdot)$ modulation operator.

M cardinality of the set $\mathcal{S}$, $|\mathcal{S}|$.

$\mathbb{N}$ set of natural numbers.

K cardinality of set $\mathcal{F}$, $|\mathcal{F}|$.

N number of vertices $|\mathcal{V}|$.

$\odot$ elementwise multiplication.

$\mathbb{P}(\Omega)$ probability of event Ω.

$\boldsymbol{\eta}$ zero-mean white Gaussian noise vector.

$\mathbf{rank(A)}$ rank of matrix $\mathbf{A}$.

$\mathbb{R}$ set of real numbers.

$\mathbf{U}_{:,\mathcal{F}}$ submatrix of $\mathbf{U}$ with columns in the set $\mathcal{F}$.

$\mathbf{x}_{\mathcal{S}}$ signal $\mathbf{x}$ restricted to the set $\mathcal{S}$.

$\mathbf{\Psi}_{\mathcal{S}}$ sampling operator $\mathcal{V} \to \mathcal{S}$.

$\mathbf{h}_n$ scaling function localized around vertex v_n.

$\mathbf{G}^*$ adjoint of matrix $\mathbf{G}$.

$\mathbf{g}_{n,r}$ spectral graph wavelet transform atom of scale r localized around vertex v_n.

$\mathbf{G}$ matrix with spectral graph wavelet atoms in the columns.

$\sigma(\cdot)$ sigmoid function.

$SV_{\max}(\cdot)$ largest singular value.

$\mathbf{U}_{\mathcal{S},\mathcal{F}}$ submatrix of Laplacian eigenvectors with rows in $\mathcal{S}$ and columns in $\mathcal{F}$.

$\mathcal{S}$ subset of graph nodes.

$\mathbf{t}_n^{\mathbf{x}}$ graph signal $\mathbf{x}$ translated to node n.

$\mathbf{u}_k$ k^{th} column of $\mathbf{U}$.

$\bar{\mathbf{u}}_n$ n^{th} row of $\mathbf{U}$.

$\mathbf{1}_N$ constant vector with all entries equal to 1.

$\mathcal{V}$ set of graph nodes.

$\mathbf{w}_{n,k}$ windowed graph Fourier transform atom localized in vertex v_n and in the k^{th} graph frequency.

$x_{n,k}^w$ windowed graph Fourier transform coefficient associated with $\mathbf{w}_{n,k}$.

$\mathbf{V}_{hh}$ weight matrix mapping the hidden state $\mathbf{h}$ to itself.

$\mathbf{W}_{hx}$ weight matrix mapping the input $\mathbf{x}$ to the hidden state $\mathbf{h}$.

Capítulo 1

Introduction

1.1 Graph Signal Processing

Many current practical problems can be modeled via data signals defined on the nodes of a weighted graph. Social media [1], transportation networks [2], wireless networks [3], genetic networks [4], and functional relationship across brain regions [5] are instances of this kind of abstract data structure. Graphs could also describe similarities between high dimensional data points in statistical learning, a capability that has been considered by some machine learning algorithms [6, 7].

Graph signal processing (GSP) extends both harmonic analysis theory and classical digital signal processing techniques in order to reveal relevant information about unstructured data by exploring the underlying topology of their domain. Some of the initial works of graph-based signal processing focused on sensor networks and wireless sensor networkss (WSNs) [8–12], which are still application areas of intensive research. Particular topics of great interest in this context are both how to recover the original WSN data from only a small subset of transmitted samples [12, 13] as well as how to design graph filters to detect anomalies in WSNs [14].

In fact, GSP has been applied to a large variety of application fields, for example, to analyze the environmental impact of burning different types of heating oil in New York City [15], to monitor the environment in smart cities [16], and to reveal mobility patterns [17]. Since functional resonance magnetic imaging (fMRI) and electroencephalography (EEG) data can be seen as signals defined over a network composed by brain regions [18, 19], GSP has been used to analyze [5, 20–24], and classify these brain signals [25], for example, finding sources of dementia in neurodegenerative diseases [26].

Most of the aforementioned applications were already tackled before the development of algorithms and frameworks based on graphs, then one may ask "Why using GSP?" The main motivation behind GSP is that the relationship between data instances can be informative about some phenomenon of interest. For example, consider an fMRI signal over a graph in which the nodes represent a set of cortical areas and the edges represent the physical connections between them. The correlation of the fMRI signal between two connected areas of the brain may have a different meaning from the correlation between two distant cortical areas, then taking the anatomical substrate into account may reveal some important information. Intuitively speaking, considering the topology of data as a prior information to algorithms can be seen as a sort of regularization and, therefore, may improve the generalization of a given algorithm by restricting its space of feasible solutions.

Another common example of GSP application is collaborative filtering for recommendation systems. Consider, for instance, the problem of recommending movies for consumers in which the ratings of some movies per person are available and one wants to predict the rating that a person would attribute for an unseen movie. If two people A and B give similar ratings for movies, then if person A likes movie C, it is likely that person B will also like it. This can be seen as a prior information and, in order to translate this problem to GSP, a graph can be built with nodes representing consumers and edges representing the similarity of ratings between each pair of people [27].

In this dissertation, a distinction is made between GSP based on the Laplacian matrix (GSP$_L$) and GSP based on the adjacency matrix (GSP$_A$), following the nomenclature of [28]. The first one relies on the structure and spectral properties of the Laplacian matrix. Although spectral methods had already been used in many data analysis/processing approaches, the first extension of theory and methods from classical signal processing to graphs using the Laplacian matrix can be attributed to the works of Antonio Ortega and David Hammond [29, 30]. The origin of GSP$_A$, on the other hand, can be referred to the development of the algebraic signal processing (ASP) [31] in 2006. ASP provided a new interpretation for signal processing based on a triple: an *algebra* of filters,[1] a *module* over the *algebra* of filters[2] and a generalization of the z-transform. This new interpretation allowed the generalization of the traditional signal processing tools to analyze and process data residing on a large variety of irregular domains. By ASP, the adjacency matrix is a natural building block for developing signal processing analysis over graphs. Although some of the GSP results in the literature relies on mathematical properties of the Laplacian matrix, there is no consensus on which is one better, GSP$_L$ or GSP$_A$, and choosing one of them depends on each application.

This dissertation employs GSP to two different applications. The first one, uses the vertex-frequency analysis (VFA), analogous to the classical time-frequency

[1] Vector space where multiplication of filters is defined.

[2] Vector space whose elements can be multiplied by the elements of the *algebra* of filters.

analysis, to detect and localize anomalous nodes in a sensor network. The second application concerns the estimation of the state of a sensor network based only on a few samples, which can be seen as a problem of simultaneously interpolating and forecasting of graph signals. To tackle this problem, the GSP-sampling theory is combined with recurrent neural networks (RNNs).

1.2 GSP Applied to Anomaly Detection

Anomalies are instances of data that significantly deviate from the common behavior. Anomalies arise in many different fields, such as frauds in credit cards, intruders in a computer network, and abnormal patterns in medical imaging. Anomalies can also occur in data residing on graphs [32]. For instance, in [33], the authors employ GSP to detect anomalies in a sensor network of weather stations in the United States. Since weather stations close to each other are expected to have similar temperature measurements, the graph signal defined by the temperature in each station is expected to be smooth in the GSP sense. Thus, the presence of malfunctioning sensor in the system can be detected by the emergence of unexpected high frequency components. Nonetheless, this approach does not provide any clue about the spatial localization of the detected fault(s). In order to localize potential anomalies in the vertex domain, Chapter 7 presents a framework in which VFA is used to extract features to be fed into an anomaly-detection algorithm.

Other works have also employed VFA to anomaly detection. In [34], graph-wavelet atoms are used to detect abnormal carpal bones. First, the graph nodes are obtained from a triangular mesh applied to each bone, then, the graph-wavelet atoms are used as features for the classification. Note that each graph corresponds to a carpal bone, thus the classification is performed in the graph level. In [14], a graph-based filtering framework was developed for anomaly detection in WSN. Basically, ideal and complementary low-pass and high-pass graph filters are designed to project data into the normal subspace (where normal data lie) and the anomalous subspace (where normal data should not lie), respectively. The cutoff frequency of the filter design is obtained by minimizing the projection error of the normal data into the normal space and the projection error of the anomalies into the anomalous space. In [35], time-varying graph signals are analyzed by graph wavelets. Authors also proposed a visual analytic tool to easily reveal interesting events from data and in [36], the aforementioned framework is extended to dynamic networks (time-varying-topology graphs) in which the classification is based on the torque of feature vectors. In order to deal with temporal information, VFA is performed on the cartesian product between the spatial graph and a path graph representing the temporal line. This construction does not scale well for both network size and timedependency range. In order to detect anomalies in time-varying graphs, a VFA-based framework is proposed in Chapter 7. Unlike [33], this specific application considers each graph node as an instance of data to be classified as anomalous or normal. Moreover, different from [35], VFA is independently applied to the historical information of the network, which could be implemented in parallel.

1.3 Joint Forecasting and Interpolation of Graph Signals

Spatiotemporal (ST) prediction is a fundamental abstract problem featuring in many practical applications, including climate analyses [37], transportation management [38], neuroscience [39], electricity markets[40], and several geographical phenomenon analyses [41]. The temperature in a city, for instance, is influenced by its location, by the season, and even by the hour of the day. Another example of data with ST dependencies is the traffic state of a road, since it is influenced by adjacent roads and also by the hour of the day. ST prediction boils down to *forecasting* (temporal prediction) and *interpolation* (spatial prediction). The former refers to predicting some physical phenomenon using historical data acquired by a network of spatially-distributed sensors. The latter refers to predicting the phenomenon with a higher spatial resolution. In this context, ST data can be seen as a network signal in which a time series is associated with each network element; the dynamics (time-domain evolution) of the time series depends on the network structure (spatial domain), rather than on the isolated network elements only. The interpolation is useful to generate a denser (virtual) network. Classical predictive models assume independence of data samples and disregard relevant spatial information [42], [43]. Vector autoregressive (VAR) [44], a statistical multivariate model, and machine learning (ML) approaches, such as support vector regression (SVR) [45] and random forest regression [46], can achieve higher accuracy than classical predictive models; yet, they fail to fully capture spatial relations.

Many artificial intelligence (AI) solutions rely on extracting the right features from a given set of data and feeding them to an appropriate machine learning algorithm. More recently, some progress has been made by applying NNs to predict ST data [37, 38, 47–50]. NNs have the capacity of not only mapping an input data to an output, but also of learning a useful representation to improve the mapping accuracy [51]. Nonetheless, the fully-connected architecture of these NNs, without any prior regularizer, may fail to extract simultaneous spatial and temporal features from data.

In order to learn spatial information from these multivariate time series, some works have combined CNNs with RNNs, such as long short term memory (LSTM) [52–56]. However, CNNs are restricted to grid-like uniformly structured data, such as images and videos. To overcome this issue, the graph convolution NNs introduced by Section 6.3 [57–60], have been used in combination with either RNN, time convolution, and/or attention mechanisms to make predictions in a variety of applications. These works are summarized in Table 8.1.

GSP theory has been applied to analyze/process many irregularly structured datasets in several applications [6, 61]. An import task addressed by GSP is interpolation on graphs, i.e., (spatially) predicting the signals on a subset of graph nodes based on known signal values from other nodes [62]. In general, graph interpolation is based on local or global approaches. Local methods, such as k-nearest neighbors [63], compute the unknown signal values in a set of network nodes using values from their closest neighbors, being computationally efficient. Global methods, on the other hand, interpolate the unknown signal values at once and can provide better results by taking the entire network into account at the expense of a higher computational burden [62, 64]. Many GSP-based interpolation techniques have been proposed [62, 65–68, 68–71]. Due to the irregular structure of graph signals, the interpolation problem may become ill-conditioned, calling for efficient selection strategies for obtaining optimal sampling sets [72, 73]. In fact, the problem of interpolating a graph signal (GS) can also be addressed as semi-supervised classification [74–78] or regression [79–81] tasks. More recently, DL solutions have also been developed [60, 82].

In many applications, affording to work with large networks may be impractical; for example, placing many electrodes at once in the human cortex may be unfeasible. Installation and maintenance costs of devices can also limit the number of sensors deployed in a network [81]. Thus, developing a predictive model capable of forecasting (temporal prediction) and interpolating (spatial prediction) time-varying signals defined on graph nodes can be of great applicability. This problem can be regarded as a semi-supervised task, since only part of the nodes are available for training. Other works have addressed this problem: in [83] the graph is extended to incorporate the time dimension and a kernel-based algorithm is used for prediction; this approach, therefore, relies on the assumption of smoothness in the time domain, which is not reasonable for many applications, such as traffic flow prediction. In [84], the ST wind speed model is evaluated in a semi-supervised framework in which only part of the nodes are used for training the model, while interpolation is performed only in test phase. Therefore, the parameters learned during the training phase do not take into account the interpolation aspect. In Chapter 8, a neural network (NN) architecture is proposed to handle ST correlations by employing GSP in conjunction with an RNN. Thus, the inherent nature of ST data is addressed by jointly forecasting and interpolating the underlying network signals. A global interpolation approach is adopted as it provides accurate results when the signal is smooth in the GSP sense, whereas an RNN forecasting model is adopted given its prior success in network prediction. Herein, not only the sampled GS is input to a predictive model but also its spectral components, which carry spatial information on the underlying graph. The major contribution of the proposed model is, therefore, the ability to learn ST features by observing a few nodes of the entire graph.

1.4 Contributions

The main contributions of this work are:

- Providing a deep review of fundamental topics of GSP such as GFT, VFA, and graph sampling-upsampling using a unified notation.
- Proposing an anomaly-detection framework based on VFA. The framework is employed to detect anomalous nodes in a time-varying graph signal of a static graph.
- Proposing a model that combines RNNs with GSP-based interpolation for jointly forecasting and interpolating time-varying graph signals. This framework is evaluated in two datasets in both supervised and semi-supervised scenarios.[3]

1.5 Publications

The aforementioned contributions were disseminated as follows:

- Lewenfus, G., Alves Martins, W., Chatzinotas, S., & Ottersten, B. (2019). On the Use of Vertex-Frequency Analysis for Anomaly Detection in Graph Signals. Anais do XXXVII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2019), 1-5.
- Lewenfus, G., Martins, W. A., Chatzinotas, S., & Ottersten, B. (2020). Joint Forecasting and Interpolation of Graph Signals Using Deep Learning. arXiv preprint arXiv:2006.01536. (Submitted to a journal.)

Further, a book on GSP (*"Processamento de Sinais sobre Grafos: Fundamentos e Aplicações", Sociedade Brasileira de Matemática Aplicada e Computacional–* in Portuguese) is currently under preparation.

[3] Code available at https://github.com/gabilew/Joint-Forecasting-and-Interpolation-of-GS.

1.6 Organization

Given the large applicability of GSP, this dissertation aims to provide a deep review on some of the fundamental aspects of GSP (Part I, covering Chapters 2-6) as well as to tackle two different real world applications (Part II, covering Chapters 7-8).

Chapter 2 starts by introducing some theoretical concepts of graph theory that will be used throughout the dissertation, including graph signal (GS), graph Fourier transform (GFT) and graph filtering. This chapter presents the distinction between GSP_L and GSP_A. Chapters 3 and 4 present the so-called *vertex-frequency analysis* (VFA), which is a graph version of the *time-frequency analysis.* In discrete signal processing (DSP), both wavelet transform and windowed Fourier transform (also termed short time Fourier transform, STFT) are well-known frequency-analysis tools that also provide localized information in time/space. GSs can also have different localized spectral properties across nodes and VFA approaches have been developed in the past decade to deal with GS [85]. The windowed graph Fourier transform (WGFT), introduced in Chapter 3, generalizes the STFT by computing the GFT of windowed GSs centered at each node of the graph. The localization analysis shows that some properties of the classical signals do not hold for GSs due to the irregularity of the underlying domain. Chapter 4 presents the *spectral graph wavelet transform* (SGWT) in which the mother wavelet kernels are designed in the spectral domain [6].

One of the main challenges of transferring signal processing approaches to graphs is that downsampling-upsampling operation on general GSs is not as straightforward as on regular signals. Chapter 5 presents strategies for sampling and interpolating bandlimited graph signals. Most of the results presented in this chapter derives from the work of Isaac Pesenson [86], which introduced the Paley Wiener space on graphs and provided theoretical substrate for extending the *Shannon-Nyquist sampling theorem* for graphs.

Due to the success of machine learning (ML), especially deep learning (DL), in a large variety of applications, it did not take so long for GSP and ML to be combined. Part II of this dissertation proposes two different approaches for problems dealing with time-varying GSs over static graphs, that is, the signal of interest dynamically changes over time whereas the topology of the data remain the same. The first approach, presented in Chapter 7, is a VFA-based framework for anomaly detection on graphs, as introduced in Section 1.2, whereas the second one, presented in Chapter 8, proposes a model combining GSP and DL to deal with the problem presented in Section 1.3. Since this second work employs a DL model, Chapter 6 provides a brief review of DL applied to multivariate time-series forecasting. Moreover, due to the growing interest in applying DL to graphs, many GSP-based neural networks have been developed in the literature to deal with different problems. Therefore, Chapter 6 also introduces the pioneer graph convolutional networks (GCNs), which aim to extend the convolutional neural networks (CNNs) to graph-structured data. A review on recent deep learning approaches tailored to graphs is addressed by [87].

It is worth mentioning that there are other signal processing concepts and tools successfully extended to deal with signals residing on graphs that are not covered by this dissertation, such as filter banks and adaptive filtering.

Parte I

Theory

Capítulo 2

Graph Signal Processing

This chapter introduces GSP as well as establishes some of the GSP notations. Section 2.1 presents the notations and definitions of graph theory that will be used throughout this theses. Section 2.2 introduces the fundamental concepts of GSP: Subsection 2.2.1 and Subsection 2.2.2 present the GFT of GSP_L and GSP_A, respectively; the convolution and filtering operations are presented in Subsection 2.2.3 and the polynomial approximation of graph filters is presented in Subsection 2.2.4. Finally, Section 2.3 presents a numerical experiment in which GSP is employed to compress GSs.

2.1 Notation and Basic Definitions

Graphs are mathematical structures that represent relations between pairs of objects. These objects are represented by a set of nodes (or vertices) $\mathcal{V} \triangleq \{v_1, \ldots, v_N\}$ and the relation between each pair of them is represented by an edge. Thus, a graph $\mathcal{G}$ is a tuple $(\mathcal{V}, \mathcal{E})$ where $\mathcal{E}$ is the set of edges connecting nodes in $\mathcal{V}$. If the relation between two nodes v_n and v_m is binary, then $\mathcal{G}$ is said to be an **unweighted graph**, otherwise it is called a **weighted graph**. Another important classification of graphs concerns the direction of the edges. If there is a relation that flows from node v_n to node v_m but not necessarily from node v_m to node v_n, the graph is said to be **directed**, otherwise, if the relations are all symmetric, the graph is said to be **undirected**. For simplicity of notation, node v_n will also be represented by its index n.

A graph can also be represented by one of the following two matrices: **incidence matrix** or **adjacency matrix**. Consider a graph $\mathcal{G}$ with N nodes and define $|\mathcal{E}| \leq N^2$ as the number of edges in $\mathcal{G}$,[1] the incidence matrix $\boldsymbol{\mathcal{I}}$ is an $N \times |\mathcal{E}|$ matrix with rows representing the vertices and columns representing the edges. If $\mathcal{I}_{21} = 1$ and

[1] Note that all the nodes in $\mathcal{G}$ are not necessarily connected to each other, then $|\mathcal{E}| \leq N^2$.

$\mathcal{I}_{31} = 1$ in an undirected graph, for example, then the edge indexed by 1 connects node v_2 and node v_3. The adjacency matrix $\mathbf{A}$, on the other hand, is an $N \times N$ matrix with rows and columns representing the input and output nodes, respectively. If $A_{23} = A_{32} = 1$, then there is an edge with weight 1 connecting nodes 2 and 3. Figure 2.1 depicts an undirected weighted graph with 6 nodes.

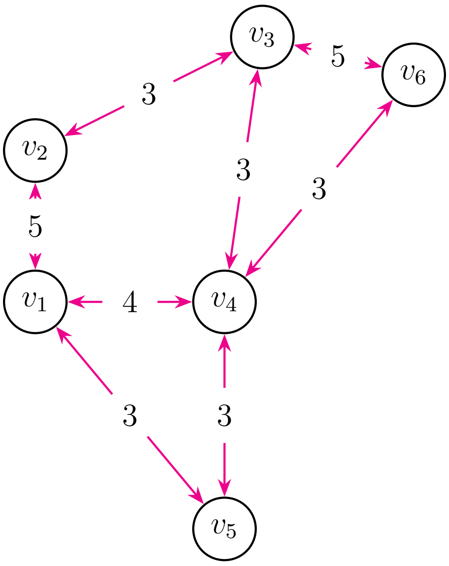


Figura 2.1: Example of an undirected graph with 6 nodes. Edges are denoted by the red arrows and edge weights are shown in the middle of the arrow. (Adapted from [88]).

The incidence and adjacency matrices associated with the graph in Figure 2.1 are given, respectively, by

$$\boldsymbol{\mathcal{I}} = \begin{bmatrix} 5 & 4 & 3 & 0 & 0 & 0 & 0 & 0 \\ 5 & 0 & 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 3 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 3 & 5 & 3 & 3 \\ 0 & 0 & 3 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 5 & 0 & 3 \end{bmatrix} \quad \text{and} \quad \mathbf{A} = \begin{bmatrix} 0 & 5 & 0 & 4 & 3 & 0 \\ 5 & 0 & 3 & 0 & 0 & 0 \\ 0 & 3 & 0 & 3 & 0 & 5 \\ 4 & 0 & 3 & 0 & 3 & 3 \\ 3 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 5 & 3 & 0 & 0 \end{bmatrix}.$$

Remark 2.1.1. *The adjacency matrix of an undirected graph is symmetric. This property will be important for the construction of the graph Fourier basis that will be described in the next section.*

2.1.1 Elements of Graph Theory

Before introducing the GSP theory, we list some concepts of graph theory that will be relevant to understand many parts of this text.

- The **degree** of node v_n, denoted as d_n, is the sum of the edge weights connecting v_n and its neighboring nodes, i.e. $d_n = \sum\limits_{m=1}^{N} A_{nm}$.

- A **path** is a sequence of edges connecting a sequence of distinct vertices. In the graph of Figure 2.1, there is a path between nodes v_1 and v_6 passing through the (finite) sequence of nodes (v_2, v_3).

- The **shortest path distance**, or path length, between vertices v_m and v_n on the graph $\mathcal{G}$, denoted as $d_{\mathcal{G}}(n,m)$, is the minimum number of edges connecting v_n to v_m. In Figure 2.1, the shortest path distance between nodes v_1 and v_6 is 2 (passing only through v_4). Note that $d_{\mathcal{G}}$ does not take edge weights into account.

- A **connected component** $\mathcal{C}$ of a undirected graph $\mathcal{G}$ is a subset of the graph $\mathcal{G}$ such that, for each pair of nodes in $\mathcal{C}$, there exists a path between them.

- An undirected graph is said to be **connected** if it is composed by only one connected component (there is a path connecting each pair of nodes in the graph).

- Let a **hop** be one unit of the shortest path distance (e.g. nodes v_1 and v_6 in Figure 2.1 are 2 hops apart, and the distance $d_{\mathcal{G}}(n,m)$ is also called hop-distance). Given a graph distance $d_{\mathcal{G}}(n,m)$, the analogous concepts of the ball and sphere from the Euclidean plane to the graph $\mathcal{G}$ are defined as follows:

  Definition 2.1.2. *The h-hop neighborhood (ball) of node v_m is the set of nodes $\mathcal{N}_h(m) \triangleq \{v_n \in \mathcal{V} \,|\, d_{\mathcal{G}}(n,m) \leq h\}$.*

  Definition 2.1.3. *The h-hop ring (sphere) around node v_m, , is the set of nodes exactly h-hops apart from node v_m, $\mathcal{N}'_h(m) \triangleq \{v_n \in \mathcal{V} \,|\, d_{\mathcal{G}}(n,m) = h\}$ or equivalently $\mathcal{N}'_h(m) = \mathcal{N}_h(m) \setminus \mathcal{N}_{h-1}(m)$.*

  For weighted graphs in which the adjacency is a matrix of similarity between pairs of nodes, it could be more appropriate to use the weighted distance

$$d^w_{\mathcal{G}}(n,m) = \min_{\mathcal{P} \in \Pi(n,m)} \sum_{e \in \mathcal{P}} I_e,$$

where

$$\Pi(n,m) \triangleq \{e_{ni_1} e_{i_1 i_2} \dots e_{i_K m} |\, \overset{e_{ni_1}}{\overset{\curvearrowright}{v_n\, v_{i_1}}}\ \overset{e_{i_1 i_2}}{\overset{\curvearrowright}{v_{i_1}\, v_{i_2}}}\ \dots\ \overset{e_{i_k m}}{\overset{\curvearrowright}{v_{i_K}\, v_m}}\ ,\ K \in \mathbb{N}\}$$

is the set of paths from n to m, $I_e = 1/A_{ij}$, and $e = e_{ij}$ is the edge connecting nodes i and j [89]. Therefore, using this distance, nodes with strong connections are closer to each other.

Assumption 2.1.4. *This dissertation assumes that all graphs are undirected and connected with no self-loops ($A_{nn} = 0 \, \forall n$).*

2.1.2 Laplacian Matrix

Another matrix that is mainly used to represent an undirected graph $\mathcal{G}$ is the Laplacian matrix. First, define the degree matrix $\mathbf{D}$ as a diagonal matrix containing each node degree d_n. The Laplacian matrix of graph $\mathcal{G}$ is defined as:

$$\mathbf{L} \triangleq \mathbf{D} - \mathbf{A}. \tag{2.1}$$

Some properties of the Laplacian matrix of undirected graphs are:

(P1) The Laplacian matrix is symmetric and has orthonormal eigenvectors;

(P2) The smallest eigenvalue is zero with multiplicity equal to the number of connected components of the graph. If the graph is connected, the multiplicity of the zero eigenvalue is 1 and it is associated with the constant eigenvector [90];

(P3) Given any pair of nodes n and m, $(\mathbf{L}^t)_{nm} = 0$ for any positive integer $t < d_{\mathcal{G}}(n, m)$. Indeed, first note that $L_{nm} \triangleq (\mathbf{L})_{nm} = 0$, if nodes v_n and v_m are not connected; then considering

$$(\mathbf{L}^t)_{mn} = \sum L_{m,p_1} L_{p_1,p_2} ... L_{p_{t-1},n}$$

with the sum taken over all the length $t-1$ sequences of graph nodes (excluding n and m), if, for contradiction, $(\mathbf{L}^t)_{nm} \neq 0$ then there would exist a sequence $p_1, p_2, ..., p_{t-1}$ such that $L_{p_i,p_{i+1}} \neq 0$, for all $1 \leq i \leq t-2$. This means that there exists a path of size at least t from n to m which contradicts the hypothesis that $t < d_{\mathcal{G}}(n, m)$.

2.2 GSP

GSP can be seen as an extension of the concepts and techniques from traditional signal processing to analyze and process signals lying on graphs. Several of the signal processing techniques rely on representing the underlying signals in a domain which makes the signal's properties under scrutiny more evident. For example, audio signals are composed by (windowed) sinusoidal waves with different frequencies and can be better analyzed in the frequency domain instead of the original time domain. The *Fourier transform*, which is able to decompose a signal into its frequency components, and its variants, such as the discrete cosine transform (DCT) and the STFT are some of the most fundamental tools in signal processing. Similarly, the first step to process signals lying on graphs is developing the concepts of frequency domain (also called spectral domain) and GFT.

A GS is a function $x : \mathcal{V} \to \mathbb{C}$ that assigns scalar values to the graph nodes, and it will be represented by the N-dimensional vector $\mathbf{x}$. To simplify, only real-valued vectors $\mathbf{x} \in \mathbb{R}^N$ will be considered in this text. It is worth mentioning that the order of arrangement of the nodes in the vector is arbitrary and does not have any impact on the frequency representation of the GS. Figure 2.2 shows an example of a GS.

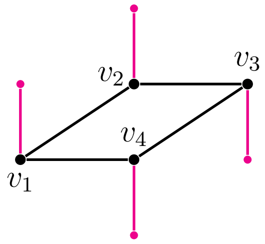


Figura 2.2: GS example for a graph with 4 nodes; in this case, $\mathbf{x} = [1\ 1\ -1\ -1]^{\mathrm{T}}$.

The GSP literature is basically divided into two approaches concerning the GFT building block: Laplacian-based GSP, (GSP$_L$) and adjacency-based GSP (GSP$_A$) [33].

2.2.1 Laplacian-based GSP

In GSP$_L$, the Laplacian eigenvectors and eigenvalues are chosen as the Fourier basis and the spectrum, respectively. The motivation behind this approach is that the Laplacian matrix, $\mathbf{L}$, can be seen as a discrete version of the Laplace-Betrami operator [91], whose eigenfunctions correspond to the Fourier basis elements in classical signal processing. Some variants of the Laplacian matrix can also be used as building block for GSP, such as the normalized Laplacian $\mathbf{L}^{\text{norm}} \triangleq \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$.

The eigenfunctions of the one-dimensional Laplacian operator, $\frac{\partial^2}{\partial t^2}$, are $\mathrm{e}^{\mathrm{j}\xi t}$, since $-\frac{\partial^2 \mathrm{e}^{\mathrm{j}\xi t}}{\partial t^2} = \xi^2 \mathrm{e}^{\mathrm{j}\xi t}$.

In DSP, the discrete Fourier transform (DFT) of a signal $\mathbf{x} \in \mathbb{R}^N$ and its respective inverse transform are

$$\hat{x}_k = \sum_{n=0}^{N-1} x_n (\omega_n^k)^* \quad \text{and} \quad x_n = \sum_{k=0}^{N-1} \hat{x}_n \omega_n^k, \tag{2.2}$$

where $\omega_n^k = \left(\mathrm{e}^{\mathrm{j}\frac{2\pi}{N}k}\right)^n$ and $[\boldsymbol{\omega}_0\ \ \boldsymbol{\omega}_1 \ldots \boldsymbol{\omega}_{N-1}]$ is the Fourier basis, with $\boldsymbol{\omega}_n =$

$[\omega_n^0 \ \omega_n^1 \ \dots \ \omega_n^{N-1}]^\mathrm{T}$.

To derive the GFT in GSP$_L$ from equation (2.2), let $\mathbf{u}_1, ..., \mathbf{u}_N$ denote the orthonormal eigenvectors of the Laplacian matrix $\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\mathrm{T}$. Denoting the rows of $\mathbf{U}$ as $\bar{\mathbf{u}}_n^\mathrm{T}$ for $n \in \{1, \dots, N\}$, the GFT of a GS $\mathbf{x}$ is given by:

$$\hat{x}_k \triangleq \sum_{n=1}^{N} x_n U_{nm} \tag{2.3}$$

or in vector notation $\hat{\mathbf{x}} \triangleq \mathbf{U}^\mathrm{T}\mathbf{x}$ (vector notation). The spectrum is given by the Laplacian eigenvalues $0 = \lambda_1 < ... \leq \lambda_N = \lambda_{\max}$. The vector $\hat{\mathbf{x}}$ in (2.3) is also called the GS frequency content. In DSP, the spectrum is ordered so that low and high frequencies are well defined. For instance, if $k = 0$, the associated $\boldsymbol{\omega}_k$ is the constant vector and if, for even N, $k = N/2$, $\boldsymbol{\omega}_k$ is a highly oscillating signal. Similar to DSP, the ordering of the graph spectrum can also be associated with the oscillation of the respective eigenvectors. Each Laplacian eigenvalue satisfies [74]:

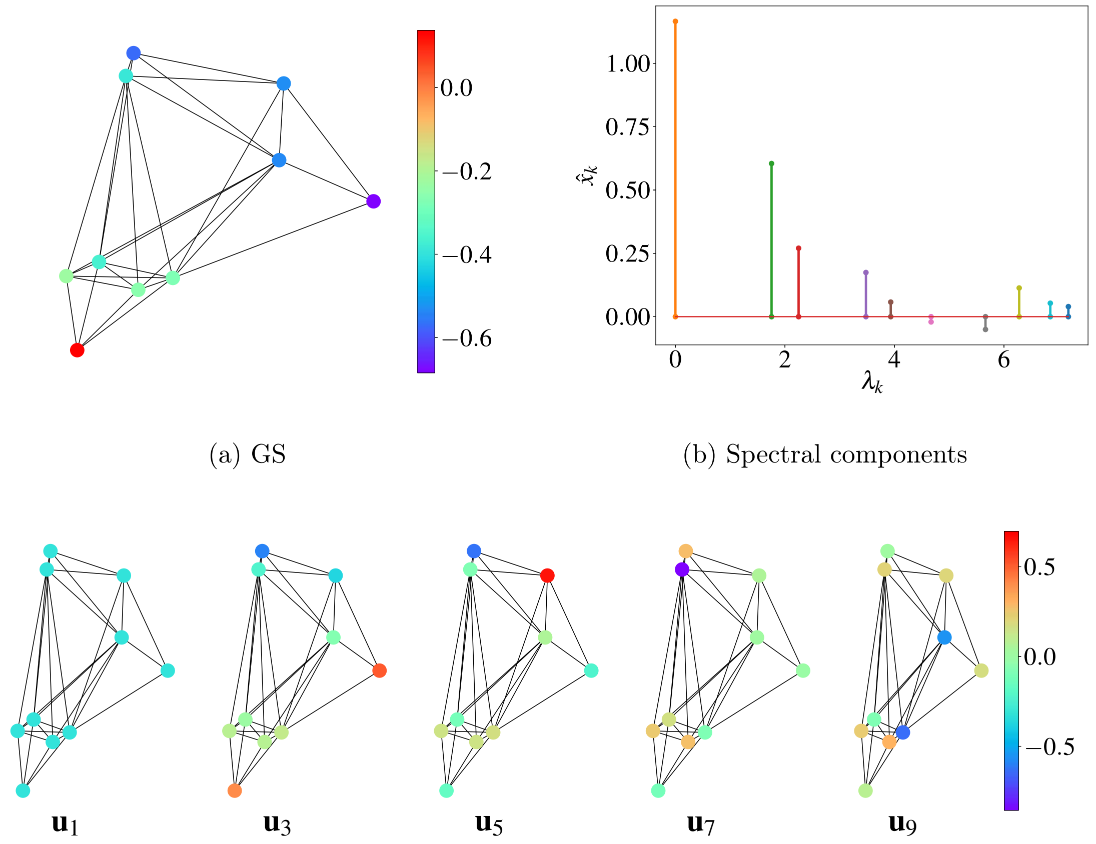
$$\lambda_k = \mathbf{u}_k^\mathrm{T}\mathbf{L}\mathbf{u}_k = \sum_{m,n} A_{mn}(U_{nk} - U_{mk})^2, \tag{2.4}$$

and, therefore, a low λ_k is associated with an eigenvector with small variations between nodes, assuming $A_{mn} \geq 0$, as shown in the following example.

Example 2.2.1. *Figure 2.3a depicts a graph with* 10 *nodes and a GS* $\mathbf{x} = \sum_{k=0}^{9} 2^{-k}\mathbf{u}_k + \boldsymbol{\eta}$*, where* $\boldsymbol{\eta}$ *is a realization of a zero mean Gaussian noise with variance* 0.01*. The frequency components of the signal* $\mathbf{x}$ *are shown in Figure 2.3b. It can be seen that the components associated with the smaller eigenvalues are larger than the components associated with the larger eigenvalues, as expected by construction. Figure 2.3c depicts some of the Laplacian eigenvectors and it can be seen that the eigenvectors oscillate faster as long as* k *increases. Note that* $\mathbf{u}_1 = \frac{1}{\sqrt{10}}\mathbf{1}$*, that is, the eigenvector associated with* $\lambda_1 = 0$ *is a constant vector.*

For specific graph structures, the same results can be obtained from GSP$_L$ and DSP. For instance, for an undirected cyclic graph as depicted in Figure 2.4, the GFT from GSP$_L$ is equivalent to the DFT, whereas for an undirected path as depicted in Figure 2.5, the GFT from GSP$_L$ is equivalent to the DCT. These two cases are explained in Example 2.2.2 and Example 2.2.3, respectively.

Example 2.2.2. *Consider the cyclic undirected path graph with six nodes as shown*



(a) GS

(b) Spectral components

(c) Laplacian eigenvectors

Figura 2.3: Example of GSP$_L$ elements. (a) depicts a GS. The GS values are represented by the colors of the graph nodes; (b) shows the spectral components of the GS in (a); and (c) shows the Laplacian eigenvectors associated with λ_1, λ_3, λ_5, λ_7 and λ_9.

in Figure 2.4. The Laplacian matrix corresponding to this graph is

$$\mathbf{L} = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & -1 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ -1 & 0 & 0 & 0 & -1 & 2 \end{bmatrix},$$

and the Laplacian eigenvalues are $\lambda_k = 2 - 2\cos\left(\frac{2\pi k}{N}\right)$. *A possible choice for orthonormal eigenvectors is* $\mathbf{u}_k = \frac{1}{\sqrt{N}}\boldsymbol{\omega}_k$, *leading to a GFT,* $\mathbf{U}^{\mathrm{T}}$, *that coincides with the normalized DFT matrix from DSP.*

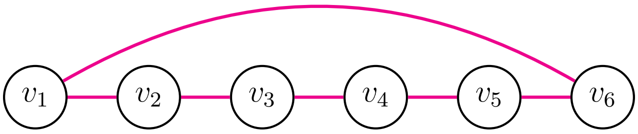


Figura 2.4: Undirected cyclic graph with $N = 6$.

Example 2.2.3. *Consider the undirected path graph with six nodes in Figure 2.5. The Laplacian matrix corresponding to this graph is*

$$\mathbf{L} = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & 0 & -1 & 1 \end{bmatrix},$$

and the Laplacian eigenvalues are $\lambda_k = 2 - 2\cos\left(\frac{\pi k}{N}\right)$. *The orthonormal eigenvectors can be chosen as* $U_{nk} = \sqrt{\frac{2}{N}}\cos\left(\frac{\pi k(n-0.5)}{N}\right)$, $k \in \{1, ..., N-1\}$, $\mathbf{u}_1 = \frac{1}{\sqrt{N}}\mathbf{1}_N$, *leading to a GFT,* $\mathbf{U}^{\mathrm{T}}$ *that coincides with the DCT-II.*

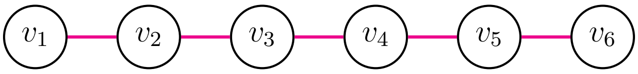


Figura 2.5: Undirected path-graph with $N = 6$.

2.2.2 Adjacency-based GSP

In GSP$_A$, the adjacency matrix is used as the shift operator (i.e. $\tilde{\mathbf{x}} = \mathbf{A}\mathbf{x}$ is the shifted signal $\mathbf{x}$). In DSP, advancing a signal x_n by m samples is the same as doing

$x_n * \delta_m$,where $*$ denotes the linear convolution operator and δ_n is the Kronecker delta function. As the Fourier transform of $h_n = \delta_{n+m}$ is $H(\mathrm{j}\xi) = \mathrm{e}^{\mathrm{j}\xi\mathrm{m}}$, the Fourier basis and spectrum in GSP_A are defined by the adjacency eigenvectors and eigenvalues, respectively. The notation for adjacency eigenpairs will be the same as Laplacian's eigenpairs, then GFT is also expressed as in (2.3). Similarly to GSP_L, the adjacency matrices of undirected graphs are symmetric and therefore has orthonormal eigenvectors and real eigenvalues. Unlike GSP_L, the adjacency eigenvalues of an undirected graph can assume any value along the real line and are not necessarily non-negatives. The GSP_A theory also encompasses directed graphs, whose adjacency matrices are not symmetric or not even diagonalizable. If the adjacency matrix is diagonalizable, the GFT operator is given by $\mathbf{U}^{-1}$.[2] If the adjacency matrix is not diagonalizable, on the other hand, it has not enough linear independent eigenvectors to compose the Fourier basis.

In order to derive a GFT for GS defined over graphs whose adjacency matrices are non-diagonalizable, the eigenvectors and the generalized eigenvectors of the adjacency matrix are concatenated in the matrix $\mathbf{V}$ such that $\mathbf{A} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1}$ and $\mathbf{J}$ is an upper triangular matrix composed by the Jordan normal blocks [92]. In this case, the GFT of a GS $\mathbf{x}$ is given by $\mathbf{V}^{-1}\mathbf{x}$ [93]. In practice non-diagonalizable matrices are not used. If the adjacency matrix $\mathbf{A}$ of a graph model is not diagonalizable, then $\mathbf{A}$ can be perturbed by ϵ sufficiently small such that $\tilde{\mathbf{A}} = \mathbf{A} + \epsilon\mathbf{I}_N$ is diagonalizable [94]. In [95], another approach, based on optimization, is developed to deal with directed graphs. In this text, only undirected graphs will be addressed, hence both the Laplacian and adjacency matrices will be symmetric and the GFT will be $\mathbf{U}^{\mathrm{T}}$.

In GSP_A it is common to normalize the adjacency matrix as $\mathbf{A}_{\text{norm}} = \frac{1}{|\lambda_{\max}|}\mathbf{A}$, where, in this case, $\lambda_{\max}$ is the eigenvalue with maximum absolute value. The normalized adjacency matrix provides more stable numerical results [33].

Unlike Laplacian spectrum, ordering frequencies in GSP_A is not straightforward, that is, the magnitudes of the adjacency eigenvalues are not correlated with the degree of oscillation of the corresponding eigenvectors. To obtain ordered frequencies in GSP_A, first consider the total variation (TV) of a GS defined by

$$\mathrm{TV}(\mathbf{x}) \triangleq \|\mathbf{A}_{\text{norm}}\mathbf{x} - \mathbf{x}\|_1 \tag{2.5}$$

The graph frequency λ_ℓ associated with the eigenvector $\mathbf{u}_\ell$ is higher than the graph frequency λ_k associated with $\mathbf{u}_k$ if $\mathrm{TV}(\mathbf{u}_\ell) > \mathrm{TV}(\mathbf{u}_k)$ [33]. The following theorem provides an ordering for the adjacency spectral elements.

Theorem 2.2.4. *Let $(\lambda_k, \mathbf{u}_k)$ and $(\lambda_\ell, \mathbf{u}_\ell)$ be two eigenpairs of the adjacency matrix.*

[2] If the adjacency matrix is symmetric, $\mathbf{U}^{-1} = \mathbf{U}^{\mathrm{T}}$.

For now, it is not necessary to consider $\mathbf{A}_{norm}$ *symmetric (thus* λ_k *and* λ_ℓ *are allowed to be complex valued). Then, for* $\|\mathbf{u}_k\|_1 = \|\mathbf{u}_\ell\|_1$, $\mathrm{TV}(\mathbf{u}_k) > \mathrm{TV}(\mathbf{u}_\ell)$ *if and only if* $|\lambda_k - |\lambda_{\max}|| > |\lambda_\ell - |\lambda_{\max}||$.

Demonstração.

$$\begin{aligned}
&\mathrm{TV}(\mathbf{u}_k) > \mathrm{TV}(\mathbf{u}_k) \iff \\
&\|\mathbf{A}_{\text{norm}}\mathbf{u}_k - \mathbf{u}_k\|_1 > \|\mathbf{A}_{\text{norm}}\mathbf{u}_\ell - \mathbf{u}_\ell\|_1 \iff \\
&\left|\frac{\lambda_k}{\lambda_{\max}} - 1\right| \|\mathbf{u}_k\|_1 > \left|\frac{\lambda_\ell}{\lambda_{\max}} - 1\right| \|\mathbf{u}_\ell\|_1 \iff \\
&\left|\frac{\lambda_k}{\lambda_{\max}} - 1\right| > \left|\frac{\lambda_\ell}{\lambda_{\max}} - 1\right| \iff \\
&|\lambda_k - |\lambda_{\max}|| > |\lambda_\ell - |\lambda_{\max}||.
\end{aligned}$$

□

Corollary 2.2.5. *If* $\lambda_k, \lambda_\ell \in \mathbb{R}$, *then* $\mathrm{TV}(\mathbf{u}_\ell) > \mathrm{TV}(\mathbf{u}_k)$ *if and only if* $\lambda_k < \lambda_\ell$.

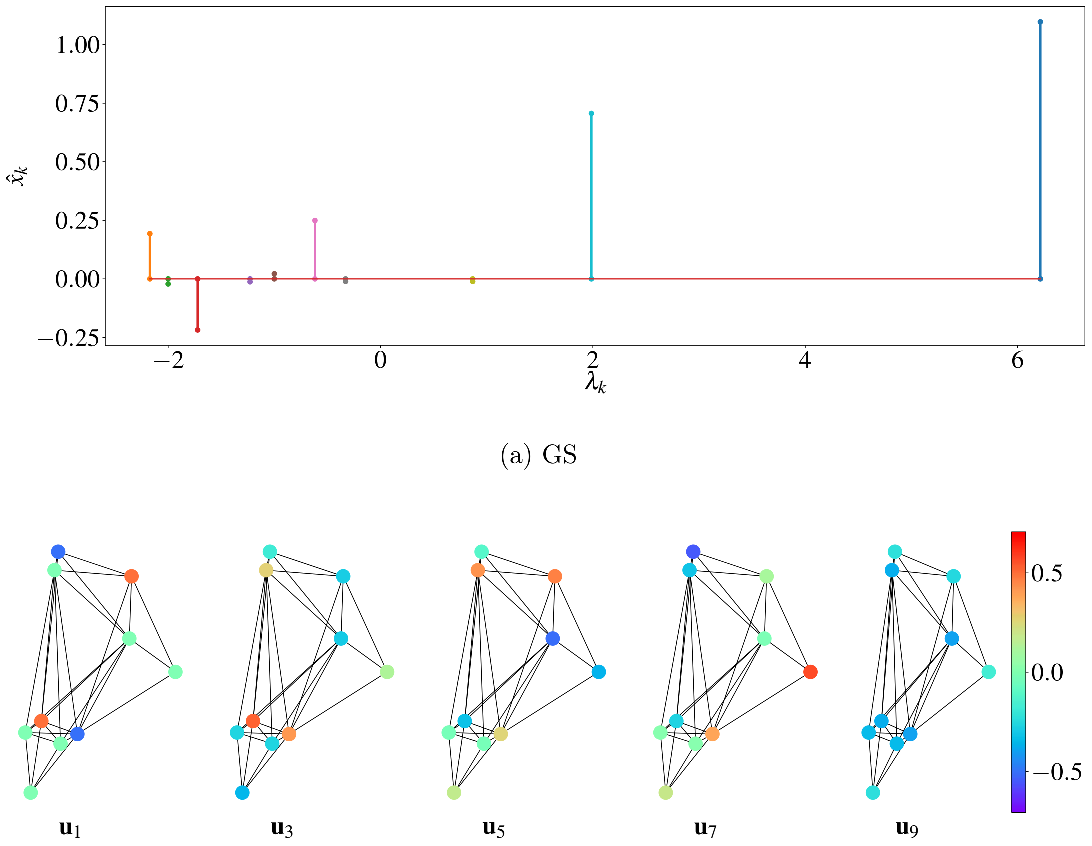
Demonstração.

$$\begin{aligned}
&\mathrm{TV}(\mathbf{u}_k) > \mathrm{TV}(\mathbf{u}_k) \iff \\
&|\lambda_k - |\lambda_{\max}|| > |\lambda_\ell - |\lambda_{\max}|| \iff \\
&-\lambda_k + |\lambda_{\max}| > -\lambda_\ell + |\lambda_{\max}| \iff \\
&\lambda_k < \lambda_\ell
\end{aligned}$$

as $|\lambda_{\max}| \geq \lambda_k \ \forall k$. □

Therefore, in GSP$_A$, contrary to GSP$_L$, the smallest λ_k's are associated with high oscillating eigenvectors whereas the largest eigenvalues are associated with smooth eigenvectors. Example 2.2.6 describes the Fourier analysis of the graph signal in Example 2.2.1 in the GSP$_A$ context.

Example 2.2.6. *Consider the graph and the GS in Figure 2.3, its Fourier coefficients are shown in Figure 2.6 and the odd indexed eigenvectors (according to an increasing order of the corresponding Laplacian eigenvalues) are shown in Figure 2.6b. It can be seen that the eigenvectors associated with smaller eigenvalues oscillate faster, as stated in Corollary 2.2.5. Moreover, the largest frequency content is associated with the smoothest eigenvector (largest magnitude eigenvalue).*



(a) GS

(b) Eigenvectors

Figura 2.6: Example of GSP$_A$ elements on the GS from Figure 2.3. (b) eigenvectors associated with $\lambda_1, \lambda_3, \lambda_5, \lambda_7$, and λ_9 depicted as graph signals.

2.2.3 Convolution and Filtering

Convolution is a very important operation in DSP since it is the mathematical operation that models linear filtering. The convolution of a continuous signal $x(t)$ with a continuous filter $h(t)$ is

$$y(t) = \int_{-\infty}^{\infty} x(\tau)h(t-\tau)\mathrm{d}\tau. \tag{2.6}$$

In DSP, the integral in (2.6) is replaced by summation:

$$y_n = \sum_{m\in\mathbb{Z}} x_m h_{n-m}. \tag{2.7}$$

The signal h is generally called **impulse response** and carries the main properties of a shift-invariant linear system. One of the main properties of convolution is the convolutional theorem which states that, under certain conditions, the convolution of two signals in the time-domain is equivalent to the element-wise multiplication of their frequency-domain representations. In DSP, the convolutional theorem can be stated as follows:

$$\mathrm{DFT}_N(\mathbf{x} * \mathbf{h}) = (\mathrm{DFT}_N\mathbf{x}) \odot (\mathrm{DFT}_N\mathbf{h}), \tag{2.8}$$

where $\mathbf{x} = [x_1\, x_2\, \ldots\, x_n]^{\mathrm{T}}$, $\mathbf{h} = [h_1\, \ldots\, h_Q]^{\mathrm{T}}$ is the vector of filter taps, and DFT_N is the normalized DFT matrix.

In both GSP_A and GSP_L, the convolution between the GS $\mathbf{x}$ and the GS $\mathbf{y}$ is inspired by the classical convolution theorem in (2.8), being defined in the frequency domain as the element-wise multiplication [30]:

$$\mathbf{x} * \mathbf{y} = \mathbf{U}(\hat{\mathbf{x}} \odot \hat{\mathbf{y}}), \tag{2.9}$$

where $\odot$ is the element-wise multiplication operator. Thus filtering is defined in the frequency domain as follows: consider the function $h : [\lambda_{\min}, \lambda_{\max}] \to \mathbb{R}$ and let $\tilde{\mathbf{H}} = h(\boldsymbol{\Lambda})$ be a diagonal matrix with diagonal entries $h(\lambda_k)$; the output signal $\mathbf{y}$ is

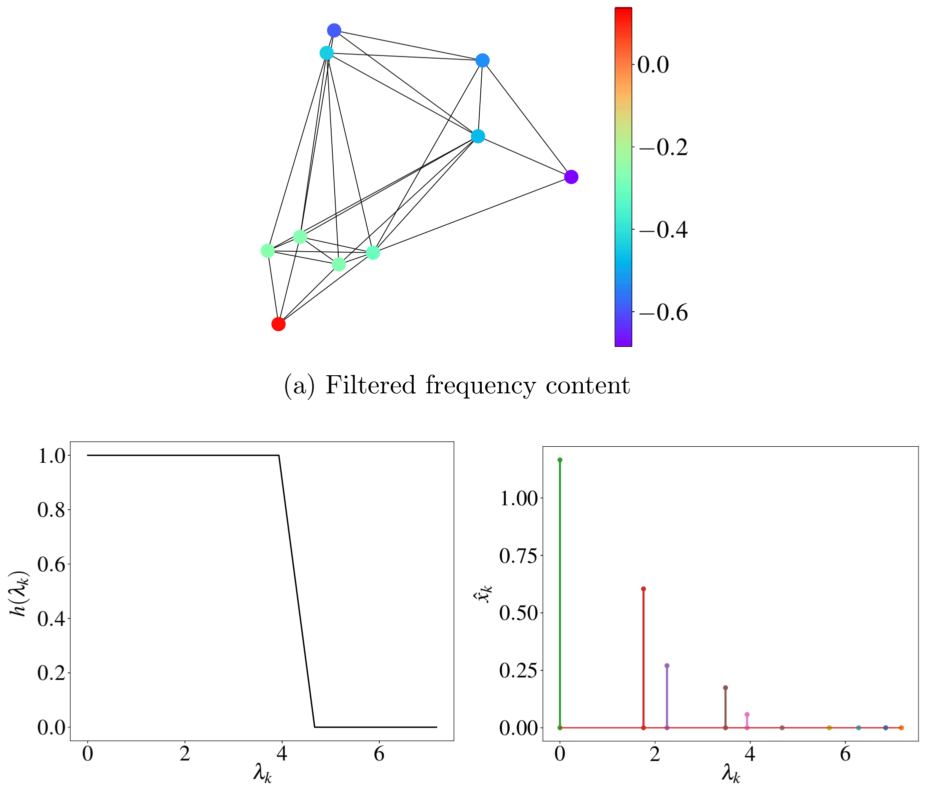
$$\mathbf{y} = \mathbf{U}\hat{\mathbf{y}} = \mathbf{U}\tilde{\mathbf{H}}\hat{\mathbf{x}} = \mathbf{U}\tilde{\mathbf{H}}\mathbf{U}^{\mathrm{T}}\mathbf{x} = \mathbf{H}\mathbf{x}, \tag{2.10}$$

in which $\mathbf{H} \triangleq \mathbf{U}\tilde{\mathbf{H}}\mathbf{U}^{\mathrm{T}}$ is the graph-domain representation of filter h. Remembering

Example 2.2.1, consider the ideal GSP$_L$ low-pass filter

$$h(\lambda) = \begin{cases} 1 & \lambda < \lambda_5 \\ 0 & \text{elsewhere.} \end{cases} \tag{2.11}$$

Figure 2.7 depicts the GS from Figure 2.3 filtered by h. The ideal low-pass filter removes the frequency content associated with Laplacian eigenvalues larger than λ_4 (high frequencies) whereas the frequency content associated with the other Laplacian eigenvalues (low frequencies) remain unchanged.



(a) Filtered frequency content

(b) Ideal low-pass filter

(c) Filtered signal

Figura 2.7: GS filtering.

Note that the low-pass filter in (2.11) requires the knowledge of λ_5. Although, filters can be designed regardless of the graph spectrum, developing graph filters in this way can lead to undesirable results due to the irregular distribution of general Laplacian (and adjacency) eigenvalues. For example, consider a graph with 4 nodes and Laplacian eigenvalues $0, 2, 2.5, 3$, and define the scalar indicator function:

$$\mathbb{1}_{[\lambda<\alpha]} = \begin{cases} 1, & \lambda < \alpha \\ 0, & \text{otherwise,} \end{cases} \tag{2.12}$$

the low-pass filter $\mathbb{1}_{[\lambda<1.5]}(\lambda)$ will vanish at all the frequencies of a GS except for the DC component $\hat{x}_1$.

2.2.4 Polynomial Graph Filters

If $\mathbf{H} = h(\mathbf{L})$ is a $(Q+1)$-order polynomial on $\mathbf{L}$, then $\mathbf{H}$ is a linear shift-invariant filter ($\mathbf{HL} = \mathbf{LH}$) when the shift of $\mathbf{x}$ on $\mathcal{G}$ is $\mathbf{Lx}$ [33]:

$$\mathbf{HL} = \left(\sum_{q=0}^{Q} c_q \mathbf{L}^q\right)\mathbf{L} = \sum_{q=0}^{Q} c_q \mathbf{L}^{q+1} = \mathbf{L}\sum_{q=0}^{Q} c_q \mathbf{L}^q = \mathbf{LH}$$

The following theorem states a necessary and sufficient condition on the graph shift $\boldsymbol{\mathcal{A}}$ and filter $\mathbf{H}$ such that $\mathbf{H}$ is linear shift invariant on $\mathcal{G}$.

Theorem 2.2.7. *(Theorem 1 in [96]) Let $\boldsymbol{\mathcal{A}}$ be the shift operator of a graph signal model such that the characteristics and minimal polynomials are equal, then the graph filter $\mathbf{H}$ is linear-shift invariant if and only if it is a polynomial in $\boldsymbol{\mathcal{A}}$.*

Polynomial filters take some advantages:

- It can be computed without accessing the entire eigendecomposition of the shift operator. To compute the graph filter $\mathbf{H}$ illustrated in Figure 2.7, for example, the Laplacian matrix was decomposed as $\mathbf{L} = \mathbf{U}\boldsymbol{\Lambda}\mathbf{U}^{\mathrm{T}}$ and the low-pass filter h was evaluated on each Laplacian eigenvalue to obtain $\mathbf{H} = \mathbf{U}h(\boldsymbol{\Lambda})\mathbf{U}^{\mathrm{T}}$.
- Filtering a GS by a Q-degree polynomial filter requires $Q|\mathcal{E}|$ operations whereas a general filter requires N^2 operations for matrix vector multiplication plus $O(N^3)$ for eigendecomposition.
- Polynomial filters are localized in the vertex domain depending on the sparsity level of the shift operator and also on the polynomial degree Q (Property 3 in Section 2.1.2). This property will be explored by VFA in Chapter 4.

Polynomial filters can be used to approximate general filters. The following example shows the implementation of an ideal low-pass filter using least squares approximation.

Example 2.2.8. *Consider a random connected sensor graph $\mathcal{G}$ with 200 nodes and whose Laplacian matrix is normalized by its largest eigenvalue. Also consider an ideal low-pass filter $h(\lambda) = \mathbb{1}_{[\lambda<\lambda_{\max}/2]}(\lambda)$. The coefficients $\{c_0, c_1, ..., c_Q\}$ of the polynomial approximation $\tilde{h}(\lambda) = \sum_{q=0}^{Q} c_q\lambda^q$ of filter h is obtained by solving the following linear system:*

$$\begin{bmatrix} 1 & \lambda_1 & \dots & \lambda_1^Q \\ 1 & \lambda_2 & \dots & \lambda_2^Q \\ \vdots & \vdots & \ddots & \vdots \\ 1 & \lambda_M & \dots & \lambda_M^Q \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_Q \end{bmatrix} = \begin{bmatrix} h(\lambda_1) \\ h(\lambda_2) \\ \vdots \\ h(\lambda_M) \end{bmatrix}, \tag{2.13}$$

where $0 = \lambda_1, \lambda_2, ..., \lambda_M = 1$ *are distinct Laplacian eigenvalues and* $Q < M \leq 200$*. The normalization of the Laplacian matrix is necessary to promote numerical stability of the matrix on the left-hand side of equation* (2.13)*. This linear system is overdetermined and can be solved by least square. Figure 2.8 shows a low-pass filter* h *approximated by polynomials* $\tilde{h}$ *of degrees* $Q = 10$ *and* $Q = 50$ *(top and bottom, respectively). The graph filter in the vertex domain is implemented as* $\tilde{\mathbf{H}} = \mathbf{U}\tilde{h}(\mathbf{L})\mathbf{U}^{\mathrm{T}}$*.*

2.2.5 Chebyshev Polynomial Approximation

Chebyshev polynomials $C_q(x)$ are defined on the interval $[-1, 1]$ and are generated by the following recursion:

$$C_q(x) = 2xC_{q-1}(x) - C_{q-2}(x),\ x \in [-1, 1]$$

Given a function f in the space of square-integrable functions in the interval $[-1, 1]$, denoted by $L_2\left([-1,1], \frac{1}{\sqrt{1+x^2}}\right)$, with respect to the measure $\frac{1}{\sqrt{1+x^2}}$, its Q-order Chebyshev approximation is given by $\tilde{f}(x) = \frac{1}{2}c_0 + \sum_{q=1}^{Q} c_q C_q(x)$, where

$$c_q = \int_{-1}^{1} \frac{C_q(x)f(x)}{\sqrt{1+x^2}} \mathrm{d}x = \frac{2}{\pi}\int_0^{\pi} \cos(q\theta) f(\cos(\theta)) \mathrm{d}\theta$$

The matrix $\mathbf{H} = h(\mathbf{L})$ can be approximated by a Chebyshev polynomial on the Laplacian. First, in order to encompass the entire Laplacian spectrum, it is necessary to map the interval $[0, \lambda_{\max}]$ into $[-1, 1]$, giving the following construction of Chebyshev polynomials on $\mathbf{L}$:

$$C_q(\mathbf{L}) = 2\overline{\mathbf{L}}C_{q-1}(\mathbf{L}) - C_{q-2}(\mathbf{L}) \tag{2.14}$$

$$\text{where } \overline{\mathbf{L}} \triangleq \frac{2}{\lambda_{\max}}\mathbf{L} - \mathbf{I},\ C_0(\mathbf{L}) \triangleq \mathbf{I},\ \text{and } C_1(\mathbf{L}) \triangleq \overline{\mathbf{L}}, \tag{2.15}$$

and the coefficients for approximating $\mathbf{H} = h(\mathbf{L})$ via

$$\widetilde{h}(\mathbf{L}) \triangleq \frac{1}{2}c_0 + \sum_{q=1}^{Q} c_q C_q(\mathbf{L}) \tag{2.16}$$

are

$$c_q \triangleq \frac{2}{\pi} \int_0^{\pi} \cos(q\theta) h\left(\frac{\lambda_{\max}}{2}(\cos\theta + 1)\right) \mathrm{d}\theta\,. \tag{2.17}$$

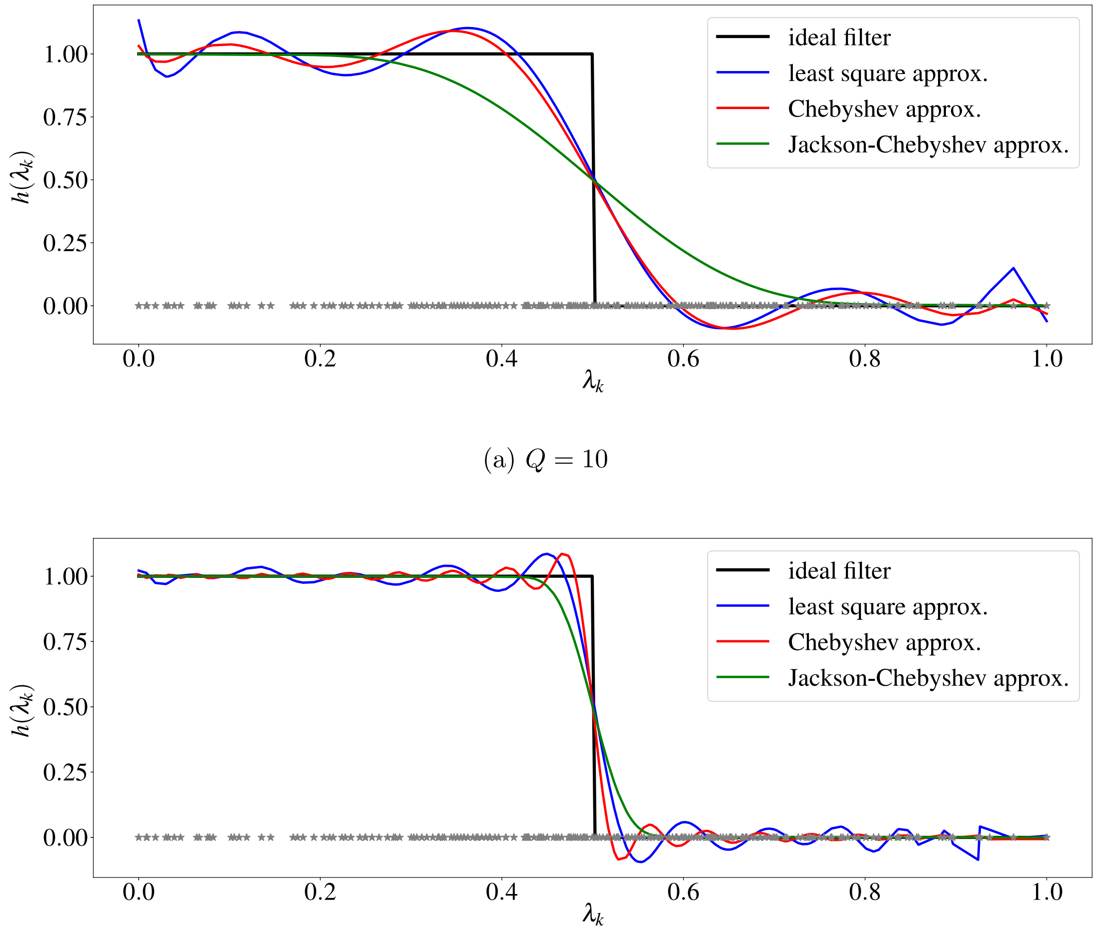
This polynomial approximation is still valid for any real symmetric matrix with $\lambda_{\min}$ not necessarily zero such as the adjacency matrix of an undirected graph. In this case, $\overline{\mathbf{L}} \triangleq \frac{2}{\lambda_{\max}-\lambda_{\min}}(\mathbf{L} - \lambda_{\min}\mathbf{I}) - \mathbf{I}$ and

$$c_q \triangleq \frac{2}{\pi} \int_0^{\pi} \cos(q\theta) h\left(\frac{\lambda_{\max}}{2}(\cos\theta + 1) + \lambda_{\min}\right) \mathrm{d}\theta\,.$$

The coefficients of the Chebyshev polynomial can be adapted to approximate the graph filter $\mathbf{H}$ by the Jackson-Chebyshev polynomial [97] as $\widetilde{\mathbf{H}} = \sum_{q=0}^{Q} a_q \mathbf{L}^q$ with $a_0 = \frac{1}{2}c_0$ and $a_q = \gamma_{q,Q} c_{s,q}$ and

$$\gamma_{q,Q} = \frac{\left(1 - \frac{q}{Q+2}\right)\sin\left(\frac{\pi}{Q+2}\right)\cos\left(\frac{q\pi}{Q+2}\right) + \frac{q}{Q+2}\cos\left(\frac{\pi}{Q+2}\right)\sin\left(\frac{q\pi}{Q+2}\right)}{\sin\left(\frac{\pi}{Q+2}\right)}. \tag{2.18}$$

This approximation is able to cancel the ripples at the cost of expanding the transition band and is useful to enhance the stop-band attenuation. The Chebyshev and the Jackson-Chebyshev polynomial approximations of the ideal low-pass filter built on the graph described in Example 2.2.8 are illustrated in Figure 2.8. In comparison with the Chebyshev polynomial approximation, the Jackson-Chebyshev polynomial approximation of degree $Q = 10$ has a larger transition band which is narrowed as long as the polynomial degree increases.



(a) $Q = 10$

(b) $Q = 50$

Figura 2.8: Ideal low pass filter and polynomial approximations. The stars represent the Laplacian eigenvalues.

2.2.6 Parseval's Identity

If the graph $\mathcal{G}$ is undirected, the Fourier basis is orthonormal, and then Parseval's identity holds for Laplacian- and adjacency-based GFT:

$$\mathbf{x}^{\mathrm{T}}\mathbf{y} = \hat{\mathbf{x}}^{\mathrm{T}}\hat{\mathbf{y}} \tag{2.19}$$

for GS $\mathbf{x}$ and $\mathbf{y}$ on $\mathcal{G}$. Since $\mathbf{U}$ is unitary, $\mathbf{x}^{\mathrm{T}}\mathbf{y} = \langle \mathbf{U}\hat{\mathbf{x}}, \mathbf{U}\hat{\mathbf{y}} \rangle = \langle \mathbf{U}^{\mathrm{T}}\mathbf{U}\hat{\mathbf{x}}, \hat{\mathbf{y}} \rangle = \hat{\mathbf{x}}^{\mathrm{T}}\hat{\mathbf{y}}$

The interpretation of Parseval's identity is that the energy of the GS is conserved when passing from vertex to frequency domain.

Most of the theoretical results presented in Chapters 3-4 assumes the eingenvalue structure of the Laplacian matrix of connected graphs.

2.3 Numerical Experiment: GS Compression

A common application of signal processing is signal compression [98, 99]. This section presents a numerical experiment of GS compression using GSP$_A$ on different graph topologies. Consider the GS from Figure 4.1. The adjacency matrix associated with this set of nodes (Brazilian weather stations) can be built as follows: the node v_n is in the neighborhood $\mathcal{N}_1(m)$ of v_m if v_n represents one of the L nearest weather stations to the station represented by node v_m or if v_m represents one of the L nearest weather stations to v_n. The resulting matrix is symmetric and its nonzero coefficients are

$$A_{mn} \triangleq \frac{\gamma_{mn}}{\sqrt{\left(\sum\limits_{i\in\mathcal{N}_n} \gamma_{in}\right)\left(\sum\limits_{j\in\mathcal{N}_m} \gamma_{jm}\right)}}, \tag{2.20}$$

where $\gamma_{mn} = \mathrm{e}^{-(d_{mn}^2+w_h h_{mn}^2)/\sigma^2}$, with d_{mn} and h_{mn} respectively denoting the geodesic distance and the height difference between stations m and n, $w_h \geq 0$ weights the contribution of the altitude, and $\sigma > 0$ is a free parameter. Note that L is the minimum number nodes connected to a single node in the graph.

For signal compression, we make the prior assumption that the GS $\mathbf{x}$ illustrated in Figure 4.1 is smooth with respect to the underlying graph $\mathcal{G}$. Since the adjacency $\mathbf{A}$ was built upon the geodesic distance between stations, close weather stations may achieve similar temperature values. Therefore, it is expected that most of the information of $\mathbf{x}$ is concentrated on a restricted spectral support $\mathcal{F} = \{N-K+1, N-K+2, ..., N\}$ associated with the K largest adjacency eigenvalues (associated with the smoothest eigenvectors) and the compressed signal $\mathbf{y}$ can be obtained as follows:

$$y_k = \begin{cases} \mathbf{u}_k^{\mathrm{T}}\mathbf{x} & k \leq K \\ 0 & \text{otherwise.} \end{cases} \tag{2.21}$$

Note that $\mathbf{y}$ is a spectral representation of the original GS $\mathbf{x}$ with loss of information. An efficient compression reduces the number of bits required to represent the data while not losing much information. In order to evaluate the loss of information, we pull the compressed signal back to the vertex domain $\tilde{\mathbf{x}} = \mathbf{U}\mathbf{y}$ and compute the error $\frac{\|\tilde{\mathbf{x}}-\mathbf{x}\|_2}{\|\mathbf{x}\|_2}$. Figure 2.9 shows the error of compression for different spectral support sizes and 4 different adjacency matrices: $L = 15$ and $w_h = 0$; $L = 15$ and $w_h = 1$; $L = 25$ and $w_h = 0$; and $L = 25$ and $w_h = 1$.

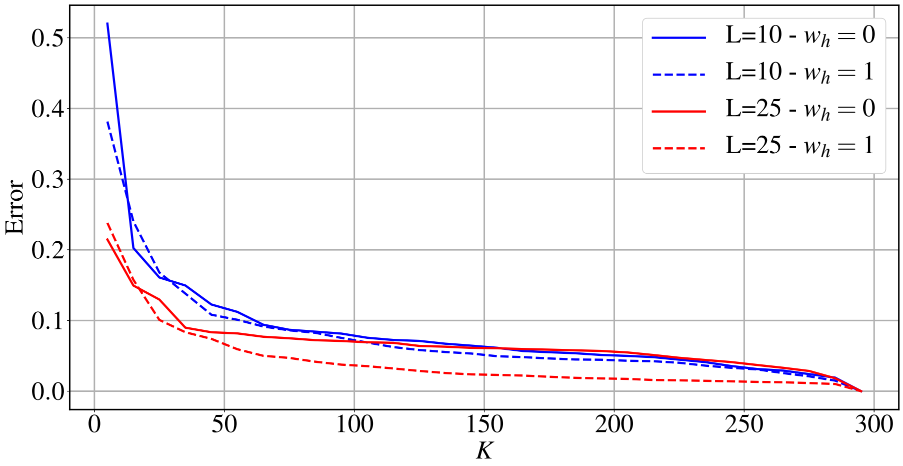


Figura 2.9: Compression error.

In Figure 2.9, the adjacency with $L = 25$ and $w_h = 1$ provided smaller reconstruction error than the other adjacency matrices. Although the latitude and the longitude can explain the temperature, the altitude can also be highly influential. Moreover, for small support size K, $L = 25$ provided better results then $L = 10$. The veracity of the smoothness assumption on the GS regards the topology of the underlying graph, thus some GSP approaches can be sensitive to the design of the adjacency or Laplacian matrices. Therefore some efforts have been made in order to optimize this design by developing algorithms able to learn the graph structure from data [100].

2.4 Conclusion

This chapter introduced the fundamentals of GSP based on both the Laplacian, GSP$_L$, or the adjacency, GSP$_A$, eigenvectors (or their variants). In GSP$_A$, the columns of the GFT are given by the adjacency eigenvectors, whereas, in GSP$_L$, the columns of the GFT are given by the Laplacian eigenvectors. The main difference between these two approaches is the ordering of the graph frequencies with respect to the ordering of the associated eigenvalues. In GSP$_L$, the eigenvectors associated with the smallest eigenvalues correspond to the low frequency Fourier elements, whereas the eigenvectors associated with the largest eigenvalues correspond to the high frequency Fourier elements. In GSP$_A$, on the other hand, the eigenvectors associated with the largest magnitude eigenvalues correspond to the low frequency Fourier elements, whereas the eigenvectors associated with the smallest eigenvalues correspond to the high frequency Fourier elements. Graph filters can be implemented in the frequency domain, as element-wise multiplication, or in the vertex domain, using polynomial approximations of the desired graph filter, such as the Chebyshev polynomial approximation, which allows the implementation of graph filters without computing the eigendecompostion of the building block matrix.

The Fourier transform is a global operation and does not capture the localized spectral pattern of non-stationary GSs. Chapters 3-4 present the WGFT and the SGWT, respectively, which are strategies to compute the frequency content of a GS locally in the vertex domain.

Capítulo 3

Windowed Graph Fourier Transform

Many real-world signals are transient and have time-varying frequency contents. Nonetheless, the Fourier transform provides a global analysis of the frequency content of a signal and does not reveal transitions in the spectral pattern. To avoid this issue and extract the frequency content of local sections of a non-stationary signal as it changes over the time, the classical STFT divides the signal into segments, via a sliding window, before applying the DFT. This tool was extended to GSs [101] as the windowed graph Fourier transform (WGFT) and was used, for example, to estimate the gyrification index of the brain [102]. In order to describe the WGFT, it is necessary to first generalize the concepts of translation and modulation.

The translation and modulation of GSs are presented in Section 3.1 and Section 3.2. Section 3.3 presents the WGFT atoms and Section 3.4 provides numerical experiments.

3.1 Translation on Graphs

In DSP, centering a signal $\mathbf{x}$ into sample n, for example, is the same as convolving $\mathbf{x}$ with the Kronecker delta signal $\boldsymbol{\delta}_n$ (vector of zeros except for the n^{th} entry, which is 1). In the GSP framework, the GFT of $\boldsymbol{\delta}_n$ is given by the n^{th} row of the eigenvector matrix, $\mathbf{U}^{\text{T}}\boldsymbol{\delta}_n = \bar{\mathbf{u}}_n$. The translation of the GS $\mathbf{x}$ to node n over a graph $\mathcal{G}$ is

$$\mathbf{t}_n^{\mathbf{x}} \triangleq \sqrt{N}\mathbf{U}\left(\hat{\mathbf{x}} \odot \bar{\mathbf{u}}_n\right). \tag{3.1}$$

There are some desirable properties of conventional translation that also hold for translations on graphs. Given the graph signals $\mathbf{x}_1$ and $\mathbf{x}_2$ over $\mathcal{G}$, the following properties of the translation operator hold:

(P1) $\mathbf{t}_n^{\mathbf{x}_1 * \mathbf{x}_2} = \mathbf{t}_n^{\mathbf{x}_1} * \mathbf{x}_2 = \mathbf{x}_1 * \mathbf{t}_n^{\mathbf{x}_2}$;

(P2) $\mathbf{t}_n^{\mathbf{t}_m^{\mathbf{x}_1}} = \mathbf{t}_m^{\mathbf{t}_n^{\mathbf{x}_1}}$;

(P3) $\sum_{m=0}^{N-1}(\mathbf{t}_n^{\mathbf{x}_1})_m = \sum_{m=0}^{N-1}(\mathbf{x}_1)_m$.

The factor $\sqrt{N}$ is included in definition (3.1) in order to preserve the DC component of the original GS. Note that property (P2) does not have the same interpretation as in the classical case. The translation of a GS to node n means concentrating $\mathbf{x}$ on node n and not necessarily shifting the $\mathbf{x}$ by n nodes across the graph. For most of the graphs, $\mathbf{t}_n^{\mathbf{t}_m^{\mathbf{x}}} \neq \mathbf{t}_{n+m}^{\mathbf{x}}$, that is, translating a GS to node m and after to node n is not equivalent to translate the original GS to node $n+m$. In the case of the regular ring and other graphs such that the respective GFTs coincide with the DFT, $\mathbf{t}_n^{\mathbf{t}_m^{\mathbf{x}}} = \mathbf{t}_{(n-1)+(m-1)-1 \bmod N}^{\mathbf{x}}$. Moreover, this definition of translation on graphs does not preserve signal energy (i.e. $\|\mathbf{t}_n^{\mathbf{x}}\|_2^2 \neq \|\mathbf{x}\|_2^2$).

The translation defined in (3.1) will be used to slide a window function, defined on the spectral domain, through a GS, therefore, in the rest of this chapter, the graph translation operator will be applied to a window vector $\mathbf{w}$ associated with a spectral graph function $w : [0, \lambda_{\max}] \to \mathbb{R}$. Common choices for w are the heat kernel $w(\lambda) = C\mathrm{e}^{-\lambda\kappa}$ and the Gaussian kernel $w(\lambda) = C\mathrm{e}^{-\kappa\lambda^2}$, where C and κ are design parameters. The goal of translating a spectral window vector $\mathbf{w}$ over a graph is to segment the GS as in the STFT, then it is important to analyze the concentration of the window $\mathbf{w}$ in the vertex-domain. The following lemma [101] provides lower and upper bounds for the energy of the translated spectral window concentrated around a node and Theorem 3.1.2 bounds the energy spread of the GS $\mathbf{t}_n^{w}$.

Lemma 3.1.1. *For any window function $w : [0, \lambda_{\max}] \to \mathbb{R}$, define the window vector $\mathbf{w} = [w(\lambda_1) \ \ldots \ w(\lambda_N)]^{\mathrm{T}}$, then*

$$w^2(0) \leq \|\mathbf{t}_n^{\mathbf{w}}\|_2^2 \leq N\mu^2\|\mathbf{w}\|_2^2, \tag{3.2}$$

where μ is the largest magnitude entry of $\mathbf{U}$, also called graph coherence.

To prove this lemma, denote the largest absolute value of $\bar{\mathbf{u}}_n$ by

$$\nu_n = \|\bar{\mathbf{u}}_n\|_\infty = \max_k |U_{nk}|$$

and the graph coherence by $\mu = \max_n \nu_n$.

Demonstração. [101] The proof of the second inequality relies on the orthogonality of $\mathbf{U}$:

$$\begin{aligned}
\|\mathbf{t}_n^{\mathbf{w}}\|_2^2 &= N\sum_{m=1}^{N}(\mathbf{t}_n^{\mathbf{w}})_m^2 = N\sum_{m=1}^{N}\left(\sum_{k=1}^{N} w(\lambda_k)U_{nk}U_{mk}\right)^2 \\
&= N\sum_{m=1}^{N}\sum_{k=1}^{N} w(\lambda_k)U_{nk}U_{mk}\sum_{\ell=1}^{N} w(\lambda_\ell)\bar{u}_{ln}U_{m\ell} \\
&= N\sum_{k=1}^{N}\sum_{\ell=1}^{N} w(\lambda_k)U_{nk}w(\lambda_\ell)U_{n\ell}\sum_{m=1}^{N} U_{mk}U_{m\ell} \\
&= N\sum_{k=1}^{N}\sum_{\ell=1}^{N} w(\lambda_k)U_{nk}w(\lambda_\ell)U_{n\ell}\delta_{\ell-k} && (3.3) \\
&= N\sum_{k=1}^{N} w^2(\lambda_k)U_{nk}^2 \leq N\sum_{k=1}^{N} w^2(\lambda_k)\nu_n^2 \leq \mu N\|\mathbf{w}\|_2^2 && (3.4)
\end{aligned}$$

where (3.3) follows from the orthonormality of the Fourier basis. To obtain the first inequality, remember that $\mathbf{u}_1 = \frac{1}{\sqrt{N}}\mathbf{1}_N$, since the Laplacian eigenvector associated with the zero eigenvalue is a constant vector. Substituting U_{n1} by $\frac{1}{\sqrt{N}}$ in (3.4):

$$\sum_{k=1}^{N} w^2(\lambda_k)U_{nk}^2 = w^2(\lambda_1) + \sum_{k=2}^{N} w^2(\lambda_k) \geq w^2(\lambda_1) = w^2(0).$$

□

If $\mu = \frac{1}{\sqrt{N}}$, then $\bar{u}_{kn}^2 = \frac{1}{N}$, $\forall k \in \{1, \ldots, N\}$, in (3.4) and $\|\mathbf{t}_n^{\mathbf{w}}\|_2 = \|\mathbf{w}\|_2$. This is the case for graphs whose GFT matrices coincide with the matrix of DFT, such as the ring graph but not for a general graph. In fact, graphs can have μ very close to 1. Table 3.1 shows the value of μ for some deterministic and random graphs.

Tabela 3.1: Coherence μ of some deterministic and random graphs. All graphs have 120 nodes, except for the Minnesota road which has 2642 nodes [103]. The regular graph has nodes with degree $d = 8$. The random clustered graph is composed by 3 clusters with nodes being connected with probability 0.5 if they are in the same cluster and 0.05 otherwise. The resulting μ of random graphs is the mean of the graph coherence obtained from 100 simulations

Deterministic	μ	Random	μ	std
Path graph	0.129	Random regular	0.347	0.022
Ring graph	0.128	Random Sensor	0.892	0.047
Comet graph	0.960	Random clustered	0.849	0.074
Minnesota road graph	0.834	-	-	-

Theorem 3.1.2. *Given nodes m and n with minimum path $d_{\mathcal{G}}(n, m) = d_{nm}$ and a d_{nm}-times continuously differentiable window function $w : [0, \lambda_{\max}] \to \mathbb{R}$ with*

$w(0) \neq 0$, *then*

$$\frac{|(\mathbf{t}_n^{\mathbf{w}})_m|}{\|\mathbf{t}_n^{\mathbf{w}}\|_2} \leq \left[\frac{2\sqrt{N}}{d_{nm}!|w(0)|}\left(\frac{\lambda_{\max}}{4}\right)^{d_{nm}}\right] \sup_{\lambda\in[0,\lambda_{\max}]} |w^{(d_{nm})}(\lambda)|. \tag{3.5}$$

(The proof is given in the Appendix A).

If $\lambda_{\max}$ is sufficiently small, the upper bound (3.5) suggests that the energy of the translated GS $\mathbf{t}_n^{\mathbf{w}}$ may be reduced in nodes faraway from node n. It is worth mentioning that the bound provided by Theorem 3.1.2 considers only the hop distance between nodes n and m and does not take edge weights into account.

3.2 Modulation on Graphs

In order to achieve localization in the frequency domain, the convolution theorem is used in its dual form: to concentrate the window GS $\mathbf{w}$ on a graph frequency λ_k, one has to multiply $\mathbf{w}$ element-wise by the Laplacian eigenvector $\mathbf{u}_k$. This operation is called modulation and is defined as

$$\mathbf{m}_k(\mathbf{w}) \triangleq \sqrt{N}\mathbf{w} \odot \mathbf{u}_k. \tag{3.6}$$

Note that, since $\mathbf{u}_1 = \frac{1}{\sqrt{N}}\mathbf{1}_N$, modulating a GS to the zero eigenvalue of a connected graph is the identity operator: $\mathbf{m}_1(\mathbf{w}) = \mathbf{w}$.

If the window vector $\mathbf{w}$ is localized around $\lambda_1 = 0$, then the modulated window $\mathbf{m}_k(\mathbf{w})$ is guaranteed to be localized around λ_k in the spectral domain as stated by the following theorem:

Theorem 3.2.1. *(Proof in [101]) Let* $w : [0, \lambda_{\max}] \to \mathbb{R}$ *be a window localized around* λ_1*. If for some* $\gamma > 0$ *the window* w *satisfies*

$$\sqrt{N}\sum_{\ell=2}^{N} \mu_\ell |w(\lambda_\ell)| \leq \frac{|w(0)|}{1+\gamma}, \tag{3.7}$$

then

$$|\widehat{\mathbf{m}}_k(\mathbf{w})_k| \geq \gamma |\widehat{\mathbf{m}}_k(\mathbf{w})_\ell| \; \forall \; \ell \neq k. \tag{3.8}$$

and

$$\frac{|\widehat{\mathbf{m}}_k(w)_k|^2}{\|\mathbf{m}_k(w)\|_2^2} \geq \frac{\gamma^2}{N-3+4\gamma+\gamma^2}. \tag{3.9}$$

Thus, the modulated window in the frequency domain $\widehat{\mathbf{m}}_k(\mathbf{w})$ is likely to localize its energy on the k^{th} spectral component. Note that Theorem 3.1.2 compares the energy on λ_k with the energy on the remaining spectral components λ_ℓ and does not take the distance in the spectral domain $\lambda_k - \lambda_\ell$ into account. When $w(0)$ is considerably larger then $w(\lambda_\ell)$, $\ell > 1$, large γ's satisfy condition (3.7) and the energy of the window w is guaranteed to be localized on k by Theorem 3.2.1. The spectral components of the modulated window kernel $w(\lambda) = C\mathrm{e}^{-\tau\lambda}$ on the path graph are shown in Figure 3.1 for $k \in \{10, 30, 50\}$ and $\tau \in \{10, 50\}$. Figure 3.2 shows the spectral components of the same modulated window kernel on the Minnesota road graph [103], a real world graph, for $k \in \{10, 1000, 2000\}$ and $\tau \in \{2, 10\}$. Note that for an irregular graph, although the peak of the modulated window occurs in λ_k, $\widehat{\mathbf{m}}_k(\mathbf{w})$ is not as concentrated as in the case of the path graph. In both examples, the larger the τ, the more concentrated the modulated window function. Nonetheless, a large τ provides poor localization on the vertex-domain. This is a case of uncertainty principle and selecting the best window width τ depends on the application.

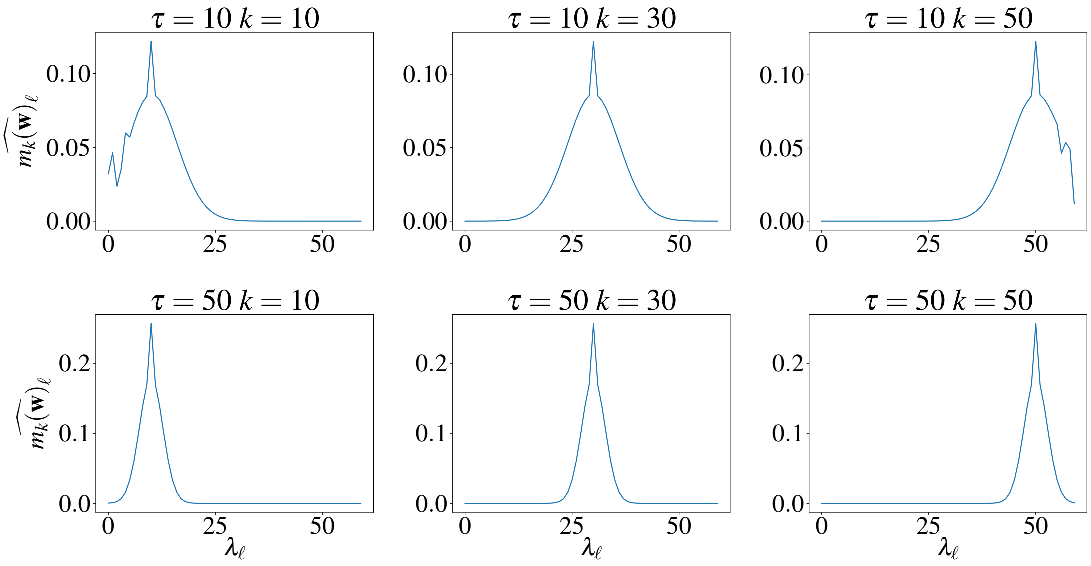


Figura 3.1: Modulation operator on the path graph. Window kernel $w(\lambda) = C\mathrm{e}^{-\tau\lambda}$.

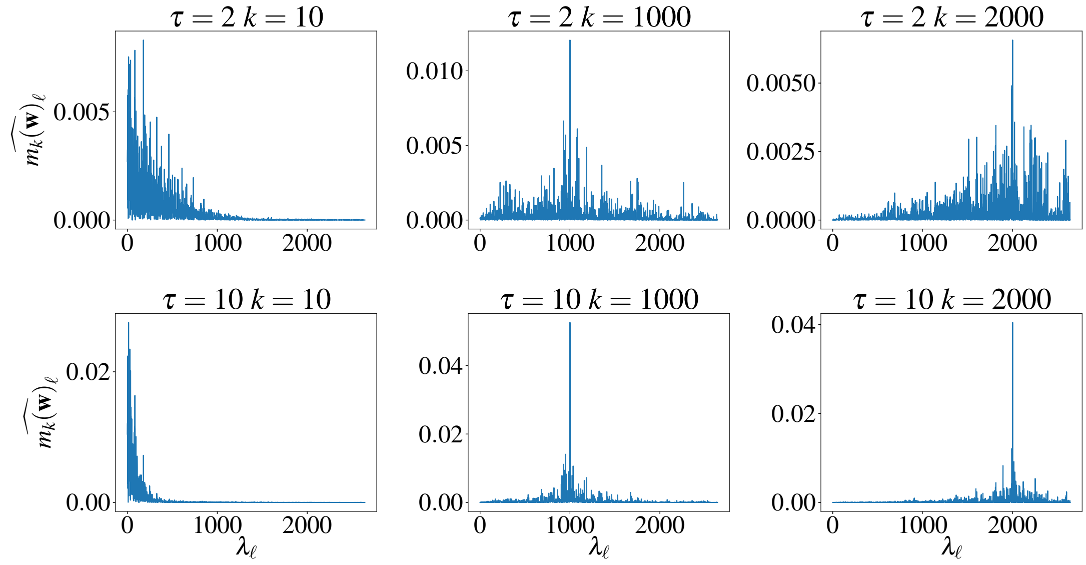


Figura 3.2: Modulation operator on the Minnesota road graph [103]. Window kernel $w(\lambda) = C\mathrm{e}^{-\tau\lambda}$.

3.3 WGFT Atoms and Spectrogram

Putting together (3.1) and (3.6), WGFT atoms and coefficients corresponding to a GS $\mathbf{x}$ are respectively defined as

$$\mathbf{w}_{n,k} \triangleq \mathbf{m}_k\left(\mathbf{t}_n^{\mathbf{w}}\right) \quad \text{and} \quad x_{n,k}^{w} \triangleq \langle \mathbf{x}, \mathbf{w}_{n,k} \rangle. \tag{3.10}$$

The matrix of WGFT coefficients $\mathbf{X}^w$ can be computed by first filtering the GS $\mathbf{x}$ by $\mathbf{T} = \mathbf{U}w(\mathbf{\Lambda})\mathbf{U}^{\mathrm{T}}$ and then multiplying each entry of $\mathbf{T}\mathbf{x}$ element-wise by each column of $\mathbf{U}$. The computational cost of this process is N^2 for matrix vector multiplication plus N^2 for element-wise multiplication, yielding an overcomplete representation composed by N^2 coefficients.

The spectrogram of the GS $\mathbf{x}$ is an $N \times N$ matrix denoted by $\mathbf{S}_{\mathbf{x}}^{w}$ with nodes represented by columns and the spectrum represented by the rows. Each entry is $(\mathbf{S}_{\mathbf{x}}^{w})_{kn} = |x_{n,k}^{w}|^2$, $n \in \{1, \dots, N\}$, $k \in \{1, \dots, N\}$. In classical signal processing, spectrograms are used to analyze the spectral pattern of a signal as it varies across time. In GSP the time domain is replaced by the vertex domain.

Figure 3.4 shows the spectrogram of a continuous by part function over a path graph with $N = 60$ nodes (top) and a clustered graph with $N = 60$ nodes (bottom). In the clustered graph, the first 20 vertices are within the first cluster, the second 20 nodes in the second cluster, and the last 20 nodes are within the last cluster. Nodes within the same cluster are connected with each other with probability $p_1 = 0.5$ whereas nodes in different clusters are connected with each other with probability

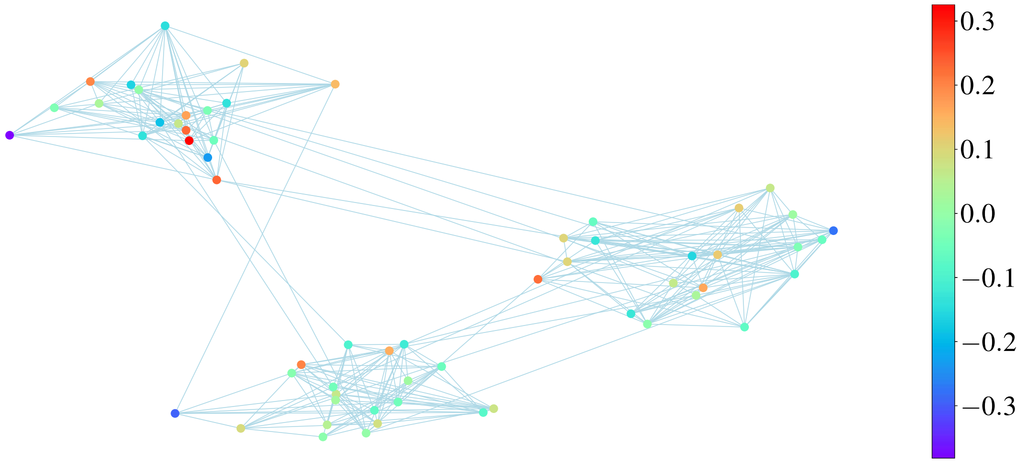


Figura 3.3: Graph with 3 clusters. The colors correspond to a continuous by part function.

$p_2 = 0.05$. This graph is shown in Figure 3.3. For both path and clustered graphs, the constructed continuous by parts signal is the combination of the eigenvector $\mathbf{u}_{15}$ on the first 20 nodes, $\mathbf{u}_{30}$ restricted to the middle nodes and $\mathbf{u}_{45}$ restricted to the last vertices. In both graphs, the window used to build WGFT atoms is $w(\lambda) = C\mathrm{e}^{-\tau\lambda}$, where constant $C = (\sum_{k=1}^{N} \mathrm{e}^{-2\tau\lambda_k})^{-1/2}$ with $\tau = 300$ in the path graph and $\tau = 5$ in the clustered graph. If τ increases, the localization in the frequency domain is improved whereas the atoms get spread in the vertex domain. However the property of vertex-frequency localization depends on the structure of the graph and may change across nodes. Indeed, given a window graph w, $\mathbf{t}_n^{\mathbf{w}}$ will be concentrated around node n but if $|U_{mk}|$ is close to zero for an $m \in \mathcal{N}_1(n)$, $\mathbf{w}_{n,k} = \mathbf{t}_n^{\mathbf{w}} \odot \mathbf{u}_k$ may not be localized around node n. Similarly, $\mathbf{m}_k(\mathbf{w})$ is localized around λ_k but if $\bar{\mathbf{u}}_{n\ell}$ is close to zero for many ℓ's close to k, then $\mathbf{w}_{n,k}$ will not be simultaneously localized around node n in the vertex domain and spectral element λ_k in the frequency domain.

If the window function w has nonzero mean, the original GS $\mathbf{x}$ is perfectly recovered from its WGFT coefficients by [101]

$$x_n = \frac{1}{\|\mathbf{t}_n^{\mathbf{w}}\|_2^2} \sum_{m=1}^{N} \sum_{k=1}^{N} x_{m,k}^{w} (w_{m,k})_n. \tag{3.11}$$

In DSP, tight frames allow the interpretation of spectrograms as an energy density function, improve stability in recovering signals from noisy measurements, and also provide faster computations [104]. It is worth pointing out that the dictionary of WGFT atoms $\{\mathbf{w}_{n,k}\}_{n,k=1}^{N}$ rarely defines a tight frame, except if $\mu = \frac{1}{\sqrt{N}}$. Appendix C briefly reviews the concept of frames and presents a theorem that provides frame bounds for the WGFT atoms [101].

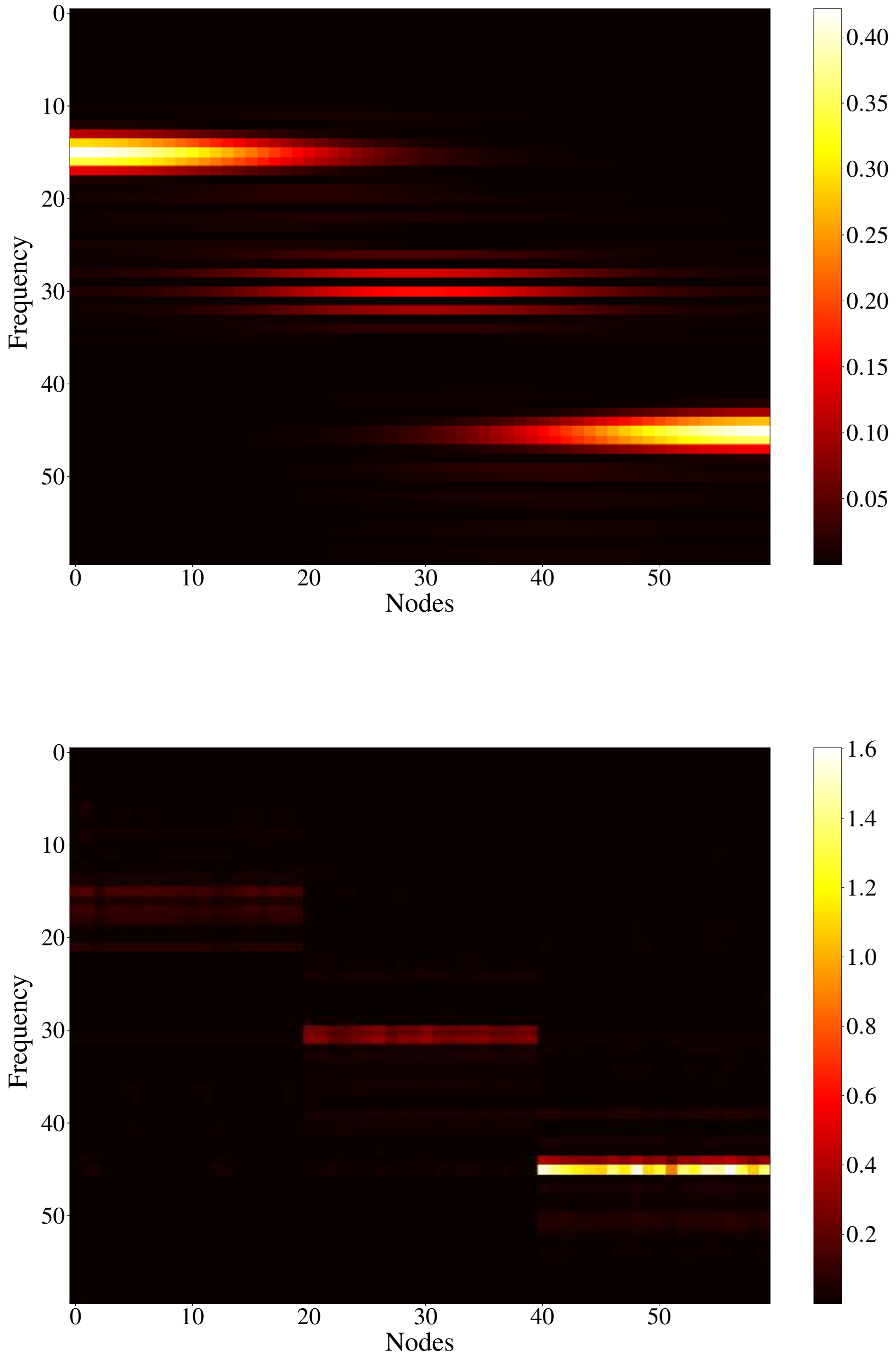


Figura 3.4: The squared magnitude of the spectrogram of the continuous by part signal on the path graph and on the 3-clustered graph in Figure 3.3.

3.4 Numerical Experiment: Spectrogram of the Brazilian Temperature Network

Consider Figure 3.5, which uses the graph structure in (2.20) to convey an artificially generated continuous by parts GS $\mathbf{x}$ defined as

$$x_n = \begin{cases} 2(\mathbf{u}_{10})_n + \epsilon, & \text{if } n \in \mathcal{N}, \\ (\mathbf{u}_{100})_n + \epsilon, & \text{if } n \in \mathcal{S}, \end{cases} \tag{3.12}$$

where $\mathcal{N}$ (north) and $\mathcal{S}$ (south) are subsets of vertices highlighted in different colors in Figure 3.5(a), and ϵ is drawn from a zero-mean Gaussian distribution with variance 0.01. The Gaussian kernel $w(\lambda) = \mathrm{e}^{-\lambda^2/10}$ is used as the spectral window for computing WGFT coefficients. Figure 3.5(b) depicts the resulting GS $\mathbf{x}$, whereas Figure 3.5(c)-(d) show the absolute values of WGFT coefficients corresponding to the two frequencies with largest coefficients' energies. As expected, nodes in the north region present larger WGFT coefficient magnitudes for frequencies around the 10^{th} Laplacian eigenvalue, whereas nodes in the south region present larger WGFT coefficient magnitudes for frequencies at the 100^{th} eigenvalue.

Many nodes in the Brazilian north region in Figure 3.5(c) achieve very small values because the eigenvector $\mathbf{u}_{11}$ is concentrated on the northeast of the graph. WGFT atoms $\mathbf{w}_{n,k}$ are not always jointly well-localized around n in vertex domain and λ_k in frequency domain, for some eigenvectors can be too much concentrated on certain vertexes. The simultaneous localization in both vertex and frequency domains is limited by the graph coherence $\mu \triangleq \max\limits_{n,k} |U_{nk}| \leq 1$. The sensor graph in Figure 3.5 has $\mu = 0.94$ (a large value) meaning that some of its WGFT atoms will not be well-localized, which is a limitation of the WGFT representation.

3.5 Conclusion

The WGFT presented in this chapter computes the GFT of a kernel window function, defined in the frequency domain, translated to each vertex of the graph. This approach has some disadvantages, for instance, the dimension of the transformed signal increases from N to N^2 and the modulation of the translated window kernel cannot be approximated by a polynomial on the Laplacian, depending on computing the Laplacian eigenvectors. Next chapter introduces the SGWT, which allows a more compact representation and can be implemented without the computation of the GFT.

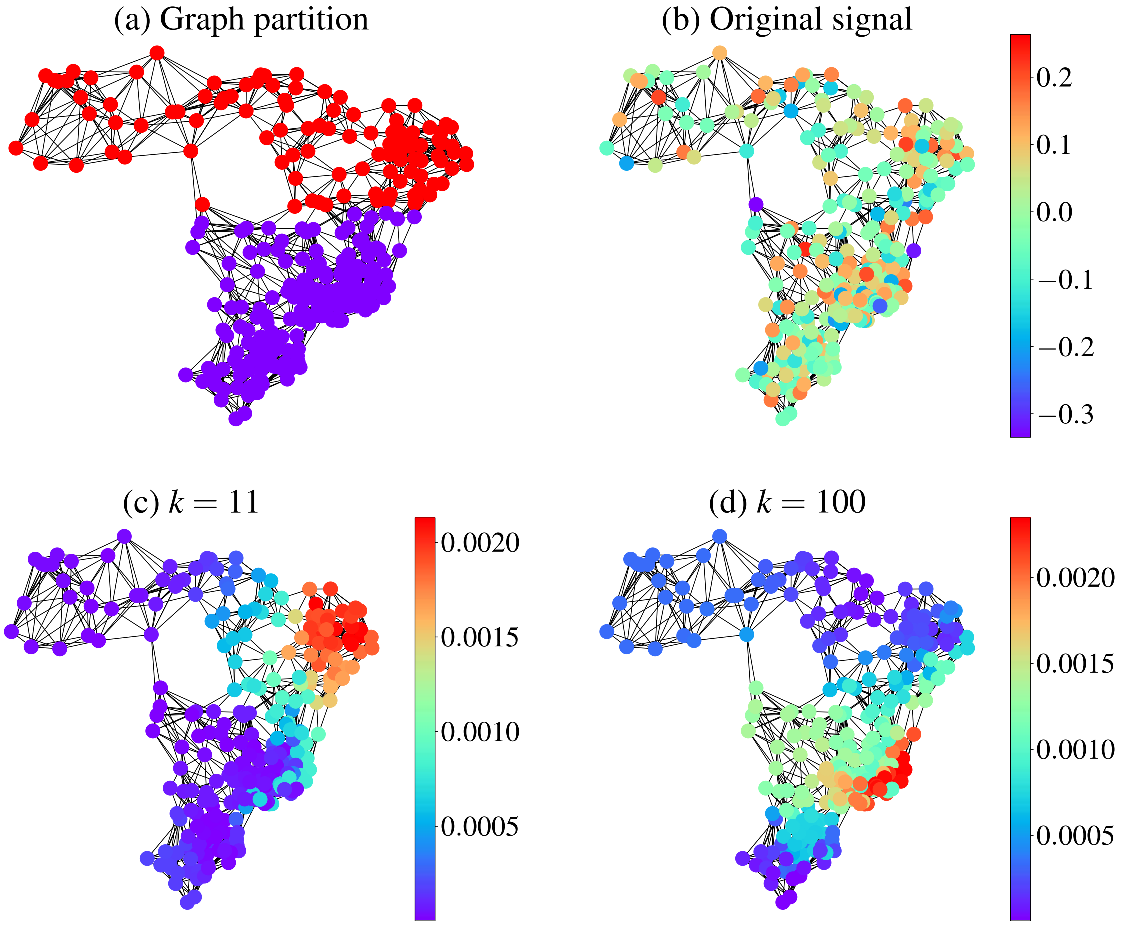


Figura 3.5: Brazilian weather station graph: (a) graph divided into two sets of nodes ($\mathcal{N}$ and $\mathcal{S}$); (b) original GS $\mathbf{x}$ in (3.12); and (c)-(d) WGFT coefficients $x^{w}_{n,k}$ in (3.10) at 11th and 100th frequencies, respectively.

Capítulo 4

Wavelets on Graphs

The WGFT introduced in the previous chapter has some disadvantages: (i) it increases the number of coefficients to represent a GS from N to N^2; (ii) it is computationally expensive and has poor scalability; indeed, WGFT atoms depend on the computation of Laplacian eigendecomposition, being prohibitive for large graphs. Actually, translation can be approximated by a Chebyshev polynomial on the Laplacian matrix (see Section 4.2), however, to the best of our knowledge, there is no method for approximating the modulation operation in (3.6); (iii) it does not provide a tight frame for irregular graphs [101]; (iv) the resolution in the frequency domain is uniform, just like in the classical DSP.

Wavelets, on the other hand, can: (i) have a more compact representation with $N \cdot (R+1)$ coefficients, with the number of scales satisfying $R < N$; (ii) be fully implemented without resorting to Laplacian eigendecomposition; (iii) provide a tight frame, depending on the choice of the mother/scale functions; (iv) have adaptive resolution in accordance with Heisenberg's theorem [104]. Therefore, this chapter introduces VFA using wavelets. Wavelets are extremely useful to extract information from data and have been used in a diverse range of applications such as data compression, denoising and transformations in general. When considering data residing on graphs, wavelet theory have also been used to extract patterns in data. The pioneers works are the diffusion wavelets [105, 106], defined on the spectral domain and the graph wavelets from [2], where authors designed a wavelet transform on graphs based on the hop distance in order to understand the behavior of a traffic network. The spectral graph wavelet transform (SGWT) [30] have been applied to pattern analysis and classification in conjunction with machine learning tools.

Although this dissertation focuses on VFA based on kernel functions built on the frequency domain, it is worth mentioning that there are also wavelet approaches developed in the vertex-domain in the literature. For instance, in [2], wavelet atoms $\mathbf{g}_{n,r}$ are constructed with respect to the r-hop neighborhood and r-hop ring around a node n and, in [107], wavelets are designed using a lifting-based scheme.

Before introducing the SGWT, we briefly review the classical wavelet theory. The classical CWT on $L_2(\mathbb{R})$[1] is generated by translating and scaling a single wavelet mother function g.

$$g_{\tau,r}(t) = \frac{1}{r} g\left(\frac{t-\tau}{r}\right),\ r > 0. \tag{4.1}$$

For a given continuous signal $x(t) \in L^2(\mathbb{R})$, the CWT is given by [108]:

$$x^g(\tau, r) = \int_{-\infty}^{\infty} x(t) g^*_{\tau,r}(t) \mathrm{d}t. \tag{4.2}$$

This CWT can be inverted if, and only if, the following admissibility condition holds

$$\int_{-\infty}^{\infty} \frac{|\hat{g}(\xi)|^2}{\xi} \mathrm{d}\xi = \Gamma_g < \infty. \tag{4.3}$$

In this case, the inverse CWT is given by [108]:

$$x(t) = \frac{1}{\Gamma_g} \int_{-\infty}^{0} \int_{-\infty}^{\infty} x^g(\tau, r) g_{\tau,r}(t) \frac{\mathrm{d}\tau \mathrm{d}r}{r} \tag{4.4}$$

The outline of this chapter is as follows: Section 4.1 presents the SGWT, the reconstruction formula, the frame analysis of the SGWT atoms, and some examples of wavelet kernels. Section 4.2 shows how to approximate the SGWT atoms by Chebyshev polynomials and Section 4.3 shows a numerical experiment in which the spectral graph wavelet transform (SGWT) is employed to recover a continuous by part GS in a semi-supervised learning (SSL) setup.

4.1 Spectral Graph Wavelet Transform

The wavelet construction in (4.1) cannot be applied to graphs, therefore, as in the WGFT definition, the SGWT is defined by a basic function, called wavelet mother/kernel, $g : \mathbb{R}^+ \to \mathbb{R}^+$ in the frequency domain [30]. This mother function must be a band-pass signal satisfying $g(0) = 0$ and $\lim_{z\to\infty} g(z) = 0$. The translation of the mother wavelet is given by element-wise multiplication in frequency domain as in the WGFT. In practice, the SGWT will be computed only for a finite set of scales α_r, $r \in \{1, 2, ..., R\}$, thus the SGWT can be defined by finite set of wavelet atoms. Wavelet atoms $\mathbf{g}_{n,r}$, with $(n, r) \in \{1, \dots, N\} \times \{1, \dots, R\}$, are GS constructed by dilating the mother function g by a factor $\alpha_r \in \mathbb{R}^+$, and then translating it to vertex

[1] The space of squared integrable functions $f : \mathbb{R} \to \mathbb{R}$.

n, as follows:

$$\mathbf{g}_{n,r} \triangleq \mathbf{U}g(\alpha_r\mathbf{\Lambda})\bar{\mathbf{u}}_n = \left(\mathbf{U}g(\alpha_r\mathbf{\Lambda})\mathbf{U}^{\mathrm{T}}\right)\boldsymbol{\delta}_n. \tag{4.5}$$

Although the mother wavelet is defined on a continuous interval, the SGWT depends on $g(\alpha_r z)$ evaluated in the Laplacian spectrum, which means that the spectral properties of the Laplacian matrix, such as eigenvalue distribution, will influence the appropriate selection of scales α_r and the general properties of the transformation.

Given a GS $\mathbf{x}$, just as in (4.2), SGWT coefficients are defined as

$$x^g_{n,r} \triangleq \mathbf{x}^{\mathrm{T}}\mathbf{g}_{n,r}\ , \tag{4.6}$$

and collecting all coefficients in a vector $\mathbf{x}^g$, one can write $\mathbf{x}^g = \mathbf{G}^{\mathrm{T}}\mathbf{x}$, where $\mathbf{G} \in \mathbb{R}^{N\times RN}$ has columns $\mathbf{g}_{n,r}$.

Wavelet atoms are orthogonal to the eigenvector $\mathbf{u}_1$ associated with $\lambda_1 = 0$,

$$\begin{aligned}\boldsymbol{g}^{\mathrm{T}}_{n,r}\mathbf{u}_1 &= \sum_{m=1}^{N}\left(\sum_{k=1}^{N} g(\alpha_r\lambda_k)U_{nk}U_{mk}\right)U_{m1} = \sum_{k=1}^{N} g(\alpha_r\lambda_k)U_{nk}\sum_{m=1}^{N} U_{mk}U_{m1}\\ &= \sum_{k=1}^{N} g(\alpha_r\lambda_k)U_{nk}\sum_{m=1}^{N}\delta_{k-1} = g(\alpha_r\lambda_1)U_{n1} = g(0)U_{1n} = 0\end{aligned}$$

where the last equality follows from $g(0) = 0$. Furthermore, the atoms are nearly orthogonal to the eigenvectors associated with the smallest eigenvalues. Therefore, in order to represent low frequencies, it is necessary to introduce a scaling function, as in classical DSP. The scaling function $h : \mathbb{R}^+ \to \mathbb{R}^+$ must be a low-pass filter, satisfying $h(0) > 0$ and $\lim_{z\to\infty} h(z) = 0$. It is worth mentioning that the scaling function is fundamental to achieve stable recovery of the original GS $\mathbf{x}$ from the SGWT coefficients, as long as scales α_r are restricted to few samples. It is also important to point out that the scaling function is introduced to the SGWT to handle the representation of low frequencies and has no impact on the choice of the wavelet mother function.

If the wavelet mother g is concentrated in the spectral domain, then the spectral graph wavelet atoms will be concentrated in each element of the Laplacian spectrum. The localization in the graph domain, on the other hand, requires a more carefully analysis. If g is sufficiently smooth in the vicinity of 0, then $\mathbf{g}_{n,r}$ will be concentrated in node n (i.e. $\frac{\mathbf{g}_{n,r}}{\|\mathbf{g}_{n,r}\|_2}$ will vanish on nodes far from n) as stated by the Theorem 4.1.1.

Theorem 4.1.1. *([30]) Let g be a (Q+1)-continuous differentiable function such that $g(0) = 0$, $g^{(q)}(0) = 0$ for all $q < Q$ and $g^{(Q)} = C \neq 0$. Suppose also that there* *exist constants r' and B such that*

$$|g^{(Q+1)}(\lambda)| < B \quad \forall \lambda \in [0, r'\lambda_{\max}],$$

then there exist constants r'' and D such that

$$\frac{g_{n,r_m}}{\|g_{n,r}\|_2} < \alpha_r D \quad \forall \alpha_r < \min(r', r'') \tag{4.7}$$

if $d_{\mathcal{G}}(n,m) > Q$.[2]

(Proof is given in Appendix A.)

4.1.1 GS Recovery

Many applications in signal processing require recovering the original signal from the transformed coefficients (i.e.: SGWT coefficients) such as denoising and compression. The SGWT admits a reconstruction formula analogous to (4.4) for continuous scales $\alpha \in \mathbb{R}_+$ [30], provided the mother wavelet g satisfies an admissibility condition equivalent to (4.3) as stated by the following theorem:

Theorem 4.1.2. *Let $g : \mathbb{R}_+ \to \mathbb{R}_+$ be a mother wavelet function, that is, $g(0) = 0$ and $\lim\limits_{z\to\infty} g(z) = 0$, also satisfying*

$$\int_0^\infty \frac{g^2(z)}{z} \mathrm{d}z < \Gamma_g \tag{4.8}$$

for some positive constant Γ_g, then

$$x_m - \hat{x}_1 U_{m1} = \frac{1}{\Gamma_g} \sum_{n=1}^{N} \int_0^\infty x^g_{n,\alpha} (g_{n,\alpha})_m \frac{\mathrm{d}\alpha}{\alpha}. \tag{4.9}$$

Theorem 4.1.2 provides a reconstruction formula for the continuous graph wavelet transform. Despite not being of practical use, since wavelet scales are generally discrete, the reconstruction formula in (4.9) provides a theoretical interesting result. Different from the classical CWT, the reconstruction formula in (4.9) recovers the centralized original signal, since $\hat{x}_1 = \sum\limits_{n=1}^{N} U_{n1} x_n = \frac{1}{N} \sum\limits_{n=1}^{N} x_n$. Therefore, only zero mean signals can be perfectly recovered.

When the scales of the SGWT are composed by a finite set $\alpha_1, ..., \alpha_r$, a signal $\tilde{\mathbf{x}}$ is recovered from the overcomplete set of SGWT coefficients $\mathbf{x}^g$ by solving the

[2] this theorem also holds for SGWT built on the adjacency eigenvectors

least-squares problem .

$$\tilde{\mathbf{x}} \triangleq \underset{\mathbf{y}}{\operatorname{argmin}} \|\mathbf{x}^g - \mathbf{G}^{\mathrm{T}}\mathbf{y}\|_2 \,, \tag{4.10}$$

over $\mathbf{y}$. It is worth mentioning that the stability of the reconstruction formula depends on some properties of the wavelet mother g and h and also on appropriate choice of scales $\alpha_1, ..., \alpha_R$. Next section discusses these properties.

4.1.2 Frame Analysis

As introduced in Chapter 3, tight frames provide more stable reconstructions, therefore, since many applications require the transformed coefficients to be pulled back to the original domain, it is important to find conditions on g and h such that the SGWT atoms provide a tight frame. Theorem 4.1.3 provides frame bounds for SGWT atoms.

Theorem 4.1.3. *([30]) Given the set of functions $\{g_r\}_{r=1}^R$ and h, the set of graph wavelets $\{\boldsymbol{g}_{n,r}\}_{n=1,r=1}^{N\;\;R} \cup \{\boldsymbol{h}_{n=1}^N\}$ is a frame with frame bounds*

$$A = \min_{\lambda\in[0,\lambda_{\max}]} G(\lambda) \tag{4.11}$$

$$B = \max_{\lambda\in[0,\lambda_{\max}]} G(\lambda) \tag{4.12}$$

where $G(\lambda) = |h(\lambda)|^2 + \sum_{r=1}^S |g_r(\lambda)|^2$.

If $G(\lambda)$ is constant for $\lambda \in [0, \lambda_{\max}]$, then $\{\boldsymbol{g}_{n,s}\}_{n=1,r=1}^{N\;\;R} \cup \{\boldsymbol{h}_{n=1}^N\}$ is tight.

(Proof is given in the Appendix C.)

4.1.3 Wavelet Functions

This section presents some scaling functions h and wavelet mothers g that have been used in the literature:

(a) cubic spline [30];

$$g(\lambda; a, b, \lambda_1, \lambda_2) = \begin{cases} \lambda_1^{-a}\lambda^a, & \lambda < \lambda_1 \\ p(\lambda), & \lambda_{\leq}\lambda \leq \lambda_2 \\ \lambda_2^b\lambda^{-b}, & \lambda > \lambda_2 \end{cases} \tag{4.13}$$

where $p(\lambda)$ is a cubic polynomial satisfying $p(\lambda_1) = p(\lambda_2) = 1$, $p'(\lambda_1) = a/\lambda_1$ and $p'(\lambda_2) = -b/\lambda_2$. Setting $a = b = 2$, $\lambda_1 = 1$ and $\lambda_2 = 2$ leads to $p(\lambda) = -5 + 11\lambda - 6\lambda^2 + \lambda^3$. Note that g is a monic polynomial for small frequencies and power decaying for high frequencies. The constraints in the definition of $p(\lambda)$ guarantee continuity of g and g' in λ.

The wavelet scales α_r are chosen to be logarithmically equispaced between $\alpha_R = \lambda_2/\lambda_{\min}$ and $\alpha_1 = \lambda_2/\lambda_{\max}$ where $\lambda_{\min} = \lambda_{\max}/K$ and and K is a design parameter. In this setting, $g(\alpha_R\lambda)$ has a monic polynomial behavior for $\lambda < \lambda_{\max}$ as desired.

The scaling function is set as $h(\lambda) = \gamma e^{-\left(\frac{\lambda}{0.6\lambda_{\max}}\right)^4}$ where $\gamma = \max_{\lambda\in[0,\lambda_{\max}]} g(\lambda)$.

(b) Meyer [109], with parameters

$$g(\lambda) = \begin{cases} \sin\left(\frac{\pi}{2}p\left(\frac{\lambda}{a}-1\right)\right), & a < \lambda \leq 2a \\ \cos\left(\frac{\pi}{2}p\left(\frac{\lambda}{2a}-1\right)\right), & 2a < \lambda \leq 4a \\ 0 & \text{elsewhere.} \end{cases} \tag{4.14}$$

$$h(\lambda) = \begin{cases} 1 & 0 < \lambda \leq a \\ \cos\left(\frac{\pi}{2}p\left(\frac{\lambda}{a}-1\right)\right), & a < \lambda \leq 2a \\ 0 & \text{elsewhere.} \end{cases} \tag{4.15}$$

where $p(\lambda) = \lambda^4(35 - 84\lambda + 70\lambda^2 - 20\lambda^3)$. Here we take $(a, M) = (\frac{2}{3}, 2)$.

(c) Hann [110], with parameters $(K, a_0, a_1, T) = (1, \frac{1}{2}, \frac{1}{2}, 3)$;

define

$$g(\lambda) = \begin{cases} \sum_{k=0}^{K} a_k \cos\left(2\pi k\left(\frac{R+1-T}{T\lambda_{\max}}\right)\right), & -\frac{T\lambda_{\max}}{R+1-T} < \lambda < 0 \\ 0 & \text{elsewhere} \end{cases} \tag{4.16}$$

with $2 < T < R$ and $K < T/2$. For each scale index $r \in \{0, \dots, R\}$ dilated wavelets are given by

$$g_r\Bigg(\lambda) = g(\lambda - r\frac{\lambda_{\max}}{R+1-T}\Bigg) \tag{4.17}$$

(d) ideal filter [111]. for $r \in \{1, ..., R-1\}$,

$$g_r(\lambda) = \begin{cases} 1, & c_r < \lambda < c_{r+1} \\ 0, & \text{elsewhere} \end{cases} \tag{4.18}$$

$$h(\lambda) = \begin{cases} 1, & 0 < \lambda < c_1 \\ 0, & \text{elsewhere} \end{cases} \tag{4.19}$$

where $c_1, ..., c_R$ are the cut frequencies of the ideal filters. The ideal filters and the Hann kernel do not follow the construction of dilatation as in the proposed SGWT definition but they were included in this section because they can be used in the same applications as the other kernels. Other SGWT mother wavelets and scaling functions are described in [109].

Figures 4.2, 4.3, and 4.4 depict the normalized SGWT coefficients of the GS in Figure 4.1 using as kernels the cubic spline, Meyer and Hann, respectively. Note that each vector of coefficients $\frac{|x_r^g|}{\sum_n |x_{n,r}^g|}$ is a GS and is depicted over the graph from Figure 4.1 instead of the real line, in order to visualize the localization in the vertex domain. Note that there are contrasting temperatures in the Northeast, resulting in high-valued coefficients for larger scales using all three kernels. The three kernels provide different representations for the same GS $\mathbf{x}$ specially due to the width of each kernel and consequently, the vertex-domain localization of the SGWT atoms.

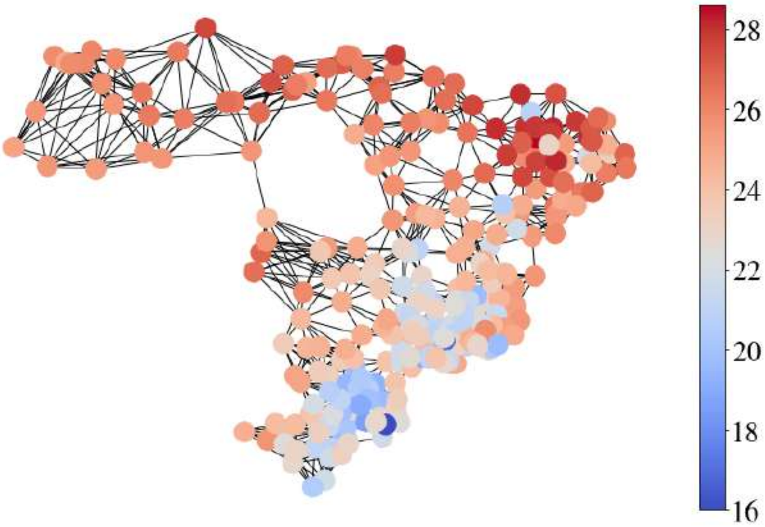


Figura 4.1: Average temperature in December on the Brazilian weather station graph.

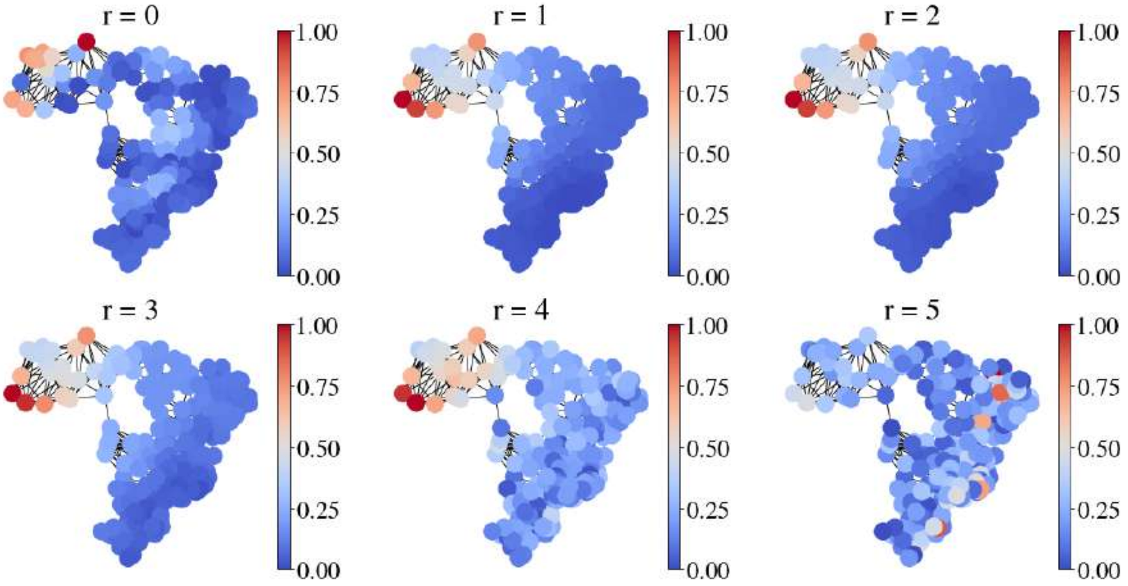


Figura 4.2: SGWT with kernel cubic spline on the Brazilian weather station graph.

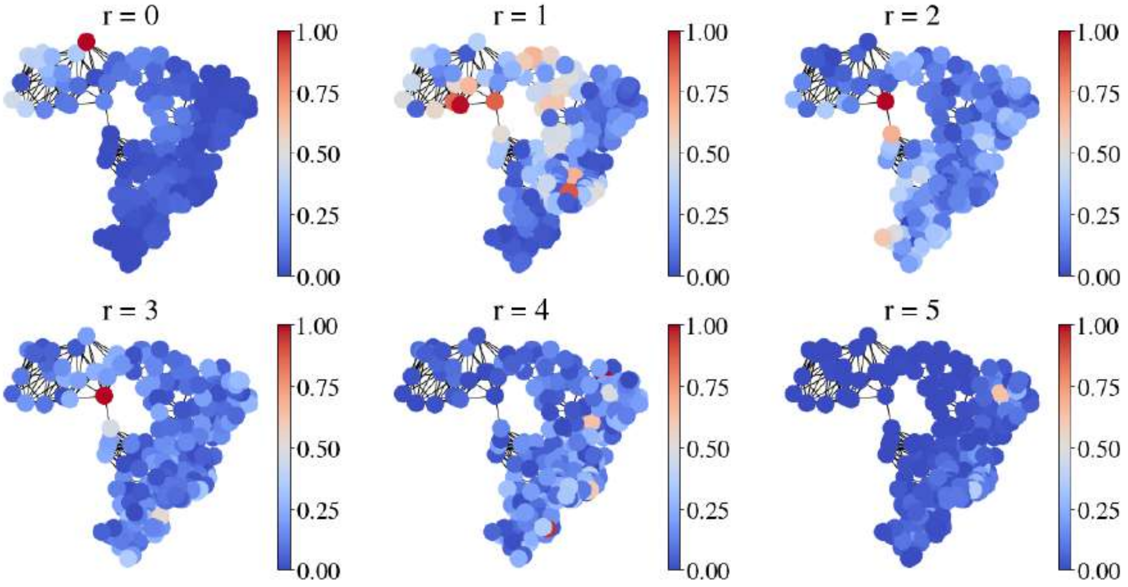


Figura 4.3: SGWT with kernel Meyer on the Brazilian weather station graph.

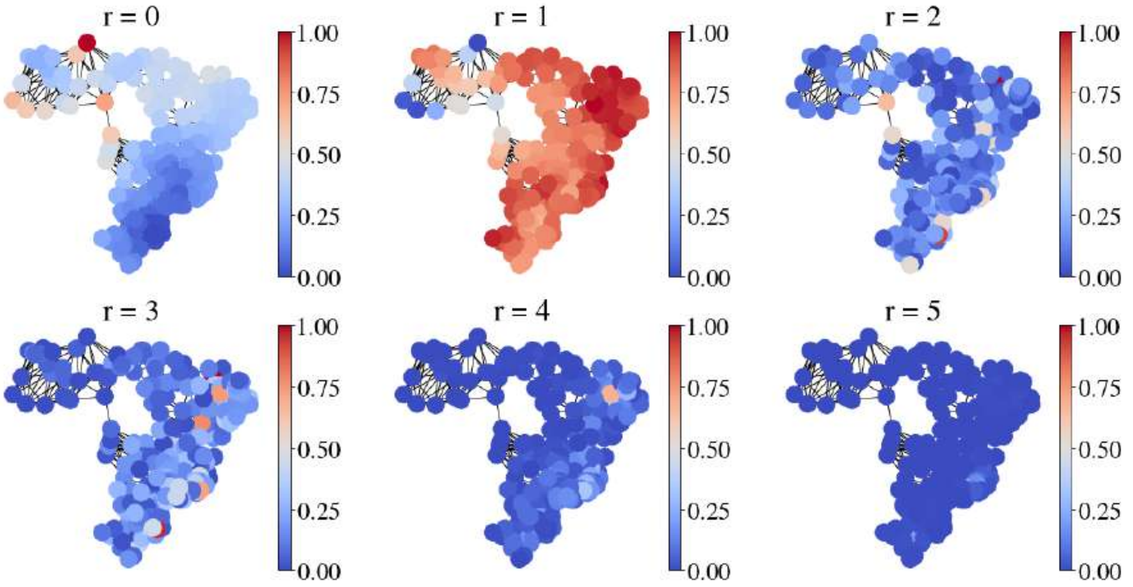


Figura 4.4: SGWT with kernel Hann on the Brazilian weather station graph.

4.2 Chebyshev Polynomial Approximation

Designing the wavelet atoms in (4.5) depends on computing the Laplacian eigendecomposition, which may be prohibitive for large graphs. Filters can actually be approximated by Chebyshev polynomials on the graph Laplacian and the filtering procedure can be performed directly in the vertex domain, as described in Section 2.2.5.

Fixing a wavelet scale r, the matrix $\mathbf{G}_r$ with $\boldsymbol{g}_{n,r}$, $n \in \{1, \ldots, N\}$, in its columns can be approximated by the Chebyshev polynomials on the Laplacian defined in (2.14). The coefficients for approximating $g_r(\mathbf{L}) \triangleq \mathbf{U} g_r(\boldsymbol{\Lambda}) \mathbf{U}^{\mathrm{T}}$ in (4.5) via

$$\widetilde{g}_r(\mathbf{L}) \triangleq \frac{1}{2} c_0 + \sum_{q=1}^{Q} c_{r,q} C_q(\mathbf{L}) \tag{4.20}$$

are

$$c_{r,q} \triangleq \frac{2}{\pi} \int_0^{\pi} \cos(q\theta) g_r \left(\frac{\lambda_{\max}}{2} (\cos\theta + 1) \right) \mathrm{d}\theta \,. \tag{4.21}$$

The Chebyshev polynomials in (2.14) allow faster computations and each $C_q(\mathbf{L})$ needs to be computed once for all filters. Only the coefficients in (4.21) need to be updated. Furthermore the computation of $\widetilde{g}_r(\mathbf{L})\mathbf{x}$ with the sum in equation (4.20) is dominated by the matrix vector multiplication $\mathbf{L}\mathbf{x}$, then the computational cost is proportional to the number of edges $|\mathcal{E}|$. Figure 4.5 depicts the approximation of cubic splines, Meyer kernels, Hann kernels, and ideal filters by a 50th-order

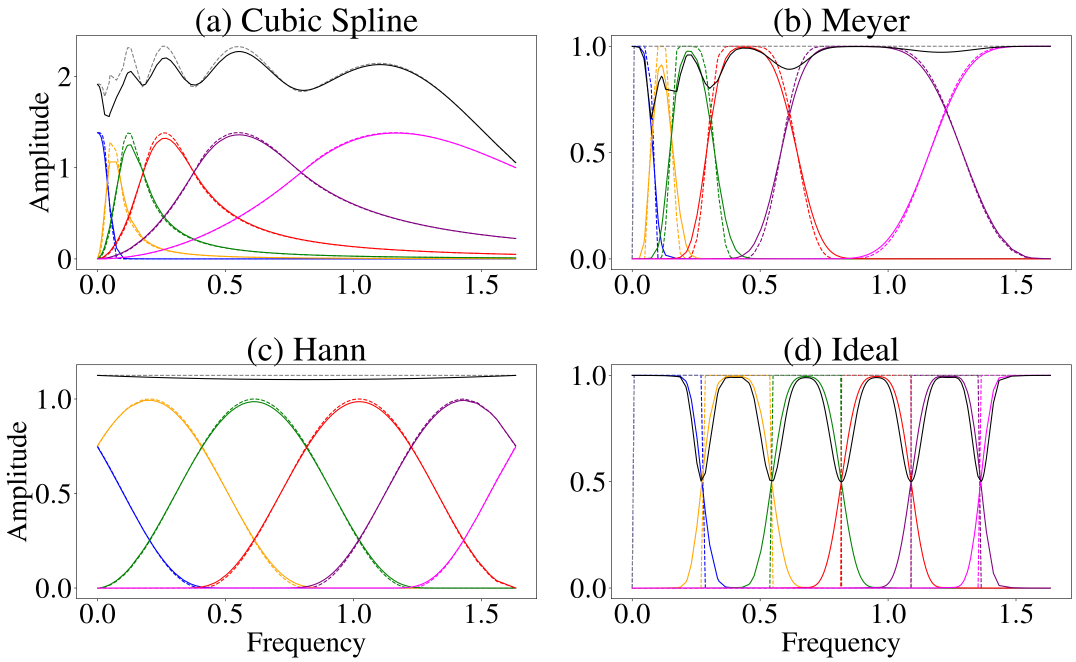


Figura 4.5: Wavelets and their approximations in frequency domain with $R = 5$. One low-pass filter (in blue) and five band-pass filters (in yellow, green, red, purple, and magenta) are shown. Dashed lines correspond to the true wavelet kernel in the range $[0, \lambda_{\max}]$. Solid lines show the corresponding 50^{th}-order polynomial approximation of the wavelet kernels: (a) cubic spline; (b) Meyer kernel; (c) Hann kernel; and (d) ideal filter. Solid black lines show $G(\lambda)$ for the approximations and dashed gray lines, for the actual wavelet kernels. The four sets of wavelet kernels were built on the Laplacian eigenvalues of the Brazilian weather station graph with adjacency entries given by (2.20).

Chebyshev polynomial. Ideal filters are approximated by the Jackson-Chebyshev polynomial [97] in (2.18).

As can be seen in Figure 4.5, the cubic spline in (a) provides higher localization for low frequencies, however, it does not generate a tight frame, since $G(\lambda)$ from Theorem 4.1.3, shown in gray dashed line, is not constant within the interval $[0, \lambda_{\max}]$, whereas Meyer, Hann, and ideal kernels do form tight frames. The Meyer wavelet [109] and the ideal filters require a high-degree polynomial to achieve good approximations. The Laplacian matrix is usually sparse but the higher the degree of approximation the denser is the approximated filters. If the graph is extremely large, a high degree approximation may be infeasible not only for matrix multiplication but also for storage. In order to have a better approximation by low-order Chebyshev polynomials, one may construct wavelet kernels based on sinusoidal waves [110, 112] as the graph wavelet based on Hann kernel shown by Figure 4.5(c). These wavelets have uniform frequency width and can be warped with a function $\omega(\lambda) = \log(\lambda)$ to achieve finest resolutions at low frequencies [110] .

4.2.1 Distributed Computation

Consider the Q-order Chebyshev polynomial approximation $\widetilde{\mathbf{H}}$ of a graph filter $\mathbf{H}$, then $\widetilde{H}_{nm} = 0$ for all node $m \notin \mathcal{N}_Q(n)$, then, the approximated SGWT coefficients can be computed locally on each node. A distributed framework is proposed in [113]. Each node n knows:

(1) The GS restricted to the neighborhood $\mathcal{N}_Q(n)$;

(2) Edge weights connecting to each of its neighbors $(A_{nm}, m \in \mathcal{N}_Q(n))$;

(3) The Q Chebyshev coefficients $c_{r,q}$ that can either be precomputed centrally and then sent to each node or be computed locally by each node;

(4) An upperbound to the Laplacian spectrum.

Each node computes $(\widetilde{\mathbf{H}}\mathbf{x})_n$ by iteratively exchanging information with local neighbors.

4.3 Numerical Experiment: Wavelets in Semi-Supervised Learning

Semi-supervised learning (SSL) is a class of machine learning techniques that can use both the labeled and the unlabeled data in training and lies between the unsupervised learning, that does not make use of any labeled data, and supervised learning, that uses only labeled data for training. The SSL is suitable for problems in which a small amount of labeled data is available in comparison with the amount of unlabeled available data. In most of the applications the labeling process is performed by a human being (i.e.: defining if a picture is a dog or a cat) and may require many time and resources.

Consider the set of data points $\mathcal{X} = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$ and the corresponding label set $\mathcal{Y} = \{y_1, ..., y_N\}$ where each $\mathbf{x}_n$ is a vector with Q attributes (numerical continuous, discrete, binary or categorical) Each y_n can also be numerical continuous, discrete, binary or categorical. The dataset $\mathcal{X}$ is composed by labeled points $\mathcal{X}_{\mathrm{L}} = \{\mathbf{x}_1, ..., \mathbf{x}_\ell\}$ (with $\ell << N$) associated with the corresponding labels $\mathcal{Y}_L = \{y_1, ..., y_\ell\}$ and by the unlabeled points $\mathcal{X}_{\mathrm{UL}} = \{\mathbf{x}_{L+1}, ..., \mathbf{x}_N\}$ whose labels $\mathcal{Y}_{UL}$ should be predicted.[3] The difference between this approach and the supervised-learning is that the unlabeled set $\mathcal{X}_{UL}$ is used in training. Assume that each pair $(\mathbf{x}_n, y_n)$ is i.i.d. and sampled from a distribution $\mathbb{P}(\mathbf{x}, y)$, then the goal of SSL is to make inference about $\mathbb{P}(y|\mathbf{x})$. The unlabeled data points, thus, provide some knowledge about $\mathbb{P}(\mathbf{x})$ that can be

[3] This class of techniques is called *transductive learning.*

useful to infer $\mathbb{P}(y|\mathbf{x})$. Typical SSL algorithms make the following assumptions [114, Section 11.6]:

1. if points $\mathbf{x}_1$ and $\mathbf{x}_2$ in a high density region are close to each other, then the corresponding labels y_1, y_2 should also be close (*smoothness assumption*);
2. if points are in the same cluster, they are likely to be in the same class (*clustering assumption*);
3. the high dimensional data lie (roughly) in a low dimensional space (*manifold assumption*).

Note that assumption (1) is more related to regression problems whereas assumption (2) is more related to classification problems. In order to translate the SSL into a GSP problem, it is necessary to define a graph and a GS. A graph $\mathcal{G}$ with nodes $\mathcal{V}$ corresponding to each data point is built using the feature vectors in $\mathcal{X}$. The Gaussian kernel $f(\mathbf{x}_n, \mathbf{x}_m) = C\mathrm{e}^{-\kappa\|\mathbf{x}_n - \mathbf{x}_m\|_2^2}$, where C and κ are design parameters, is commonly used to compute adjacency matrix entries A_{mn} but other functions such as cosine similarity can also be used. It is worth mentioning that, in order to satisfy conditions 1 and 2 of SSL, the adjacency matrix should be a similarity matrix such that nodes with strong connections should be in the same class (1) and corresponding labels should be close to each other (2). The resulting adjacency matrix can be thresholded to promote sparsity. The GS is a vector $\mathbf{y} \in \mathbb{R}^N$ with labels y_ℓ in its entries. The *smoothness assumption* is equivalent to requiring smoothness of $\mathbf{y}$, that is, the energy of $\mathbf{y}$ is concentrated on low frequencies in the spectral domain. This assumption is related to the Definition 5.1.2 (bandlimited GS) introduced in Chapter 5.

Smooth GSs are also more aligned with SGWT atoms $\mathbf{g}_{n,r}$ corresponding to small frequencies, therefore, a smooth GS is expected to have relatively small SGWT coefficients $x^g_{n,r}$, $n \in \{1, \dots, N\}$, $r = \{1, \dots, R\}$ for r close to R.

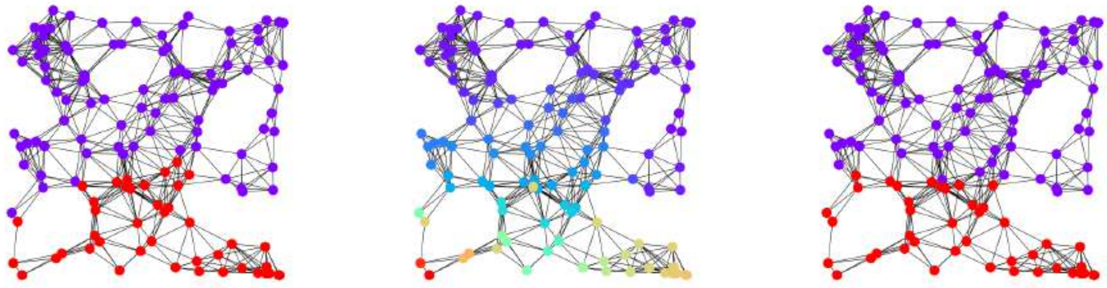
The multiresolution analysis provided by the SGWT can be used to label a piecewise continuous GS $\mathbf{x}$ in an SSL setup [74]. One method to determine the labels for the unlabeled data is to incorporate the sparsity on the wavelet coefficients:

$$\min_{\mathbf{x}^g} \frac{1}{2}\|\mathbf{y} - \mathbf{\Psi}\mathbf{G}^*\mathbf{x}^g\|_2^2 + \sum_{r=1}^{R} \gamma_r \|\mathbf{x}^g_r\|_1 \tag{4.22}$$

where $\mathbf{y} \in \mathbb{R}^N$ is a vector with labels y_n at all entries corresponding to $\mathcal{Y}_L$ zero elsewhere; $\mathbf{\Psi}$ is an $N \times N$ matrix with $\Psi_{nn} = 1$, if $\mathbf{x}_n \in \mathcal{X}_L$, zero elsewhere; $\mathbf{G}^*$ is the adjoint of the SGWT operator; and $\mathbf{x}^g = [\mathbf{x}^h, \mathbf{x}^g_1, ..., \mathbf{x}^g_R] \in \mathbb{R}^{NR}$ is the vector of wavelet coefficients. One problem in this approach is that the support of high frequency atoms may highly intersect the vertices associated with unlabeled data. Thus most of the coefficients corresponding to high frequency atoms are set to zero and the optimization problem fails to approximated the boundaries of piecewise linear functions. For example, consider the following semi-supervised classification problem: let $\mathcal{G}$ denote a random sensor graph [103] with 100 nodes as shown by Figure 4.6a. Consider the label GS $\mathbf{x}$ given by the sign of the Fiedler vector:

$$x_n = \begin{cases} 1 & \text{if } U_{n2} \geq 0 \\ -1 & \text{otherwise.} \end{cases} \tag{4.23}$$

Suppose that only a percentage $p = 0.15$ of the labels on the graph sensor is available. Figure 4.6b shows the resulting prediction, $\tilde{\mathbf{x}}$, obtained solving (4.22) with $\gamma_r = 0.2$, $r = 1, 2, 3, 4$ and the cubic spline kernel and Figure 4.6c shows the binary result sign$(\tilde{\mathbf{x}})$. As a drawback promoted by the sparsity of SGWT coefficients, nodes around the discontinuity of the original GS were misclassified.



(a) Original GS

(b) Resulting GS

(c) Resulting binary GS

Figura 4.6: Semi-supervised classification over a sensor graph.

The capability of the SGWT of dealing with the both the *smoothness* and *clustering assumptions* will be further explored in Chapter 7 in order to build both supervised and unsupervised anomaly detection frameworks.

4.4 Conclusion

The SGWT is a useful tool to analyze and process vertex-varying GSs and it can be efficiently implemented without computing the Laplacian eigenvectors. The property of localization of the SGWT atoms depends on the magnitude of the GFT entries, which are related to the topology of the graph. The use of SGWT in anomaly detection will be further explored in Chapter 7. Next chapter presents downsampling and upsampling operators for GSs, which are other fundamental tools of signal processing.

Capítulo 5

Sampling on Graphs

Downsampling and upsampling are common operations in multirate signal processing; yet it is not straightforward how to generalize those operations for GSP, since nodes are not ordered. Downsampling a discrete time signal by a factor 2 , for example, means dropping every odd sample, but an “odd node"is not a well-defined concept on a general graph. Because of this difficulty, a large variety of graph sampling approaches have been developed focusing on many different applications, such as filterbanks, active learning, semi-supervised learning (SSL), data interpolation, etc [61].

Section 5.1 presents the theory of sampling/interpolating GS based on the bandlimited assumption [68]. This section also presents theoretical results for approximated bandlimited GS and sampling strategies to optimize the reconstruction error in the presence of noise or modeling mismatch. Section 5.2 presents an interpolation method based on the cutoff frequency of the bandlimited GS.

5.1 Sampling Bandlimited Signals

In classical signal processing, Shannon-Nyquist sampling theorem states that a sampled bandlimited signal can be perfectly recovered by a sinc[1] interpolation if the sampling rate is at least twice the bandwidth.

Theorem 5.1.1. *(Sampling theorem, [115]) Let $f(t)$ be a continuous square-integrable function such that $\hat{f}(\xi) = 0$ for all $|\xi| > \xi_m$. If the sampling frequency $\xi_s \geq 2\xi_m$, then $f(t)$ can be recovered by the following interpolation formula*

$$f(t) = \sum_{n=-\infty}^{\infty} f\left(\frac{n\pi}{\xi_s}\right) \frac{\sin(\xi_s t)}{\xi_s t}. \tag{5.1}$$

[1] The sinc function is defined as $\text{sinc}(x) = \frac{\sin(x)}{x}$.

The set of functions $\{f(t) \in L^2(\mathbb{R});\ \hat{f}(\xi) = 0\ \forall \xi, |\xi| > \xi_s\}$ (square-integrable bandlimited functions) is called a Paley-Wiener space denoted by $\mathrm{PW}_{\xi_s}(\mathbb{R})$.

In order to extend sampling theory to graph signals, let $\mathcal{S} \triangleq \{s_1, ..., s_M\} \subset \mathcal{V}$ be a subset of nodes with $M \leq N$ nodes; the vector of measurements $\mathbf{x}_\mathcal{S} \in \mathbb{R}^M$ is given by $\mathbf{x}_\mathcal{S} = \mathbf{\Psi}_\mathcal{S}\mathbf{x}$, where the sampling operator

$$[\mathbf{\Psi}_\mathcal{S}]_{mn} = \begin{cases} 1, & \text{if } v_n = s_m \\ 0, & \text{otherwise} \end{cases} \tag{5.2}$$

selects from $\mathcal{V}$ the nodes in $\mathcal{S}$. The interpolation operator $\mathbf{\Phi}_\mathcal{S}$ is an $N \times M$ matrix such that the recovered signal is $\tilde{\mathbf{x}} = \mathbf{\Phi}_\mathcal{S}\mathbf{\Psi}_\mathcal{S}\mathbf{x}$. If $\tilde{\mathbf{x}} = \mathbf{x}$, the pair of sampling and interpolation operators $(\mathbf{\Phi}_\mathcal{S}, \mathbf{\Psi}_\mathcal{S})$ can perfectly recover the signal $\mathbf{x}$ from its sampled version. As the rank of $\mathbf{\Phi}_\mathcal{S}\mathbf{\Psi}_\mathcal{S}$ is smaller or equal to M, this is not possible for all $\mathbf{x} \in \mathbb{R}^N$ when $M < N$.

In order to develop a graph sampling method based on Theorem 5.1.1, it is first necessary to define a concept of bandlimited GS.

Definition 5.1.2. *(Bandlimited GS) The GS* $\mathbf{x}_\mathrm{b}$ *is said* $\mathcal{F}$*-bandlimited if* $(\hat{\mathbf{x}}_\mathrm{b})_n = 0\ \forall n$ *such that* $\lambda_n \notin \mathcal{F} \subset \{\lambda_1, \ldots, \lambda_N\}$*, that is, the frequency content of* $\mathbf{x}_\mathrm{b}$ *is restricted to the set of frequencies* $\mathcal{F}$*. The space of* $\mathcal{F}$*-bandlimited GS on the graph* $\mathcal{G}$ *with GFT* $\mathbf{U}$ *will be denoted as* $\mathrm{BL}_\mathcal{F}(\mathbf{U})$*.*

Some works also restrict the support of the frequency content and consider that a GS $\mathbf{x}_\mathrm{b}$ is K-bandlimited if $(\hat{\mathbf{x}}_\mathrm{b})_k = 0$ and $0 < k \leq K$ (the columns of $\mathbf{U}$ are ordered so that the associated eigenvalues are increasingly sorted, in the case of GSP_L, or decreasingly sorted, in the case of GSP_A) or ξ-bandlimited if $(\hat{\mathbf{x}}_\mathrm{b})_k = 0\ \forall k$ such that $\lambda_k > \xi$ [29]. In the second case, the space of ξ-bandlimited GS, $\mathrm{BL}_\xi(\mathbf{U})$, is the graph equivalent of the $\mathrm{PW}_\xi(\mathbb{R}^\mathrm{N})$ from classical signal processing. Theoretical results for ξ-bandlimited GS will be presented in Section 5.2.

Given the set of bandlimited GSs $\mathrm{BL}_\mathcal{F}(\mathbf{U})$, designing a pair of sampling and interpolation operators $(\mathbf{\Phi}_\mathcal{S}, \mathbf{\Psi}_\mathcal{S})$ relies on finding a subset of nodes $\mathcal{S} \subset \mathcal{V}$ able to keep all the information about any GS $\mathbf{x} \in \mathrm{BL}_\mathcal{F}(\mathbf{U})$.

Definition 5.1.3. *(Uniqueness set) Let* $\mathcal{S}$ *be a subset of* $\mathcal{V}$ *and* $\mathbf{x}, \mathbf{y} \in \mathrm{BL}_\mathcal{F}(\mathbf{U})$ *two GSs such that* $\mathbf{x}_\mathcal{S} = \mathbf{y}_\mathcal{S}$ *(the two GSs restricted to the subset of nodes* $\mathcal{S}$ *are equal). If* $\mathbf{y} = \mathbf{x}$*, then* $\mathcal{S}$ *is called a uniqueness set for* $\mathrm{BL}_\mathcal{F}(\mathbf{U})$*.*

In this text, a bandlimited signal is a sparse vector in the GFT domain as in Definition 5.1.2. The following theorem guarantees the perfect reconstruction of an $\mathcal{F}$-bandlimited GS for some sampling sets.

Theorem 5.1.4. *([68]) If the sampling operator $\mathbf{\Psi}_{\mathcal{S}}$ satisfies*

$$\operatorname{rank}(\mathbf{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}}) = |\mathcal{F}| = K, \tag{5.3}$$

then $\mathbf{x}_{\mathrm{b}} = \mathbf{\Phi}_{\mathcal{S}}\mathbf{\Psi}_{\mathcal{S}}\mathbf{x}_{\mathrm{b}}$ as long as $\mathbf{\Phi}_{\mathcal{S}} = \mathbf{U}_{:,\mathcal{F}}\mathbf{\Sigma}$, with $\mathbf{\Sigma}$ satisfying $\mathbf{\Sigma}\mathbf{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}} = \mathbf{I}_K$ and $\mathbf{U}_{:,\mathcal{F}}$ a submatrix of $\mathbf{U}$ with columns restricted to the indices associated with the frequencies in $\mathcal{F}$.

The condition in (5.3) is also equivalent to

$$\mathrm{SV}_{\max}(\mathbf{U}_{\overline{\mathcal{S}},\mathcal{F}}) \leq 1, \tag{5.4}$$

where $\mathrm{SV}_{\max}(.)$ stands for the largest singular value [116] and $\overline{\mathcal{S}} = \mathcal{V}/\mathcal{S}$. This means that no $\mathcal{F}$-bandlimited signal over the graph $\mathcal{G}$ is supported on $\overline{\mathcal{S}}$.

In order to have $\mathbf{\Sigma}\mathbf{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}} = \mathbf{I}_K$ we must have $M \geq K$, since $\operatorname{rank}(\mathbf{U}_{:,\mathcal{F}}) = K$. If $M \geq K$, $\mathbf{\Sigma}$ is the pseudo-inverse of $\mathbf{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}}$ and the interpolation operator is

$$\mathbf{\Phi} = \mathbf{U}_{:,\mathcal{F}}(\mathbf{U}_{:,\mathcal{F}}^{\mathrm{T}}\mathbf{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}})^{-1}\mathbf{U}_{\mathcal{S},\mathcal{F}}^{\mathrm{T}}\,. \tag{5.5}$$

Before the proof of Theorem 5.1.4, note that since $\mathbf{U}$ is non-singular, there is always at least a subset $\mathcal{S}$ such that the condition in (5.3) is satisfied. Nonetheless, for many choices of $\mathcal{S}$, $\mathbf{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}}$ can be full rank but ill-conditioned, leading to large reconstruction errors, especially in the presence of noisy measurements or in the case of approximately bandlimited GS (presented in Section 5.1.1). To overcome this issue, optimal sampling strategies, in the sense of minimizing reconstruction error, can be employed [68]. Note that $\mathbf{\Phi}$ depends on both $\mathcal{S}$ and $\mathcal{F}$, but this dependence is omitted in order to simplify the notation.

Demonstração. To prove Theorem 5.1.4 it is necessary to show that (i) the columns of $\mathbf{\Phi}$ spans $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$ and (ii) $\mathbf{x} = \mathbf{\Phi}\mathbf{\Psi}_{\mathcal{S}}\mathbf{x}$ for all $\mathbf{x} \in \mathrm{BL}_{\mathcal{F}}(\mathbf{U})$ meaning that $\mathbf{\Phi}\mathbf{\Psi}_{\mathcal{S}}$ is a projection operator on $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$. Therefore, if the original GS $\mathbf{x}$ already belongs to $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$, $\mathbf{\Phi}\mathbf{\Psi}_{\mathcal{S}}$ is an identity operator.

(i) first, by condition (5.3), $\operatorname{rank}(\mathbf{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}}) = K$, and by definition, $\operatorname{rank}(\mathbf{\Sigma}\mathbf{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}}) = K$, which means that $\mathbf{\Sigma}$ spans $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$. Therefore, $\mathbf{\Phi} = \mathbf{U}_{:,\mathcal{F}}\mathbf{\Sigma}$ also spans $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$. To prove (ii), $\mathbf{\Phi}\mathbf{\Psi}_{\mathcal{S}}$ is a projection operator, it is sufficient to show that it is an idempotent matrix:

$$(\mathbf{\Phi}\mathbf{\Psi}_{\mathcal{S}})^2 = \mathbf{\Phi}\mathbf{\Psi}_{\mathcal{S}}\mathbf{\Phi}\mathbf{\Psi}_{\mathcal{S}} = \mathbf{U}_{:,\mathcal{F}}\mathbf{\Sigma}\mathbf{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}}\mathbf{\Sigma}\mathbf{\Psi}_{\mathcal{S}} = \mathbf{U}_{:,\mathcal{F}}\mathbf{I}_K\mathbf{\Sigma}\mathbf{\Psi}_{\mathcal{S}} = \mathbf{\Phi}\mathbf{\Psi}_{\mathcal{S}}.$$

□

5.1.1 Approximately Bandlimited GS

In practice, most GSs are only approximately bandlimited [76]. A GS is approximately $(\mathcal{F}, \epsilon)$-bandlimited if [116]

$$\mathbf{x} = \mathbf{x}_\mathrm{b} + \boldsymbol{\eta}, \tag{5.6}$$

where $\mathbf{x}_\mathrm{b}$ is an $\mathcal{F}$-bandlimited GS and $\boldsymbol{\eta}$ is an $\overline{\mathcal{F}}$-bandlimited GS such that $\|\boldsymbol{\eta}\|_2 < \epsilon$. If signal $\mathbf{x}$ is sampled on the subset $\mathcal{S}$ and recovered by the interpolator in (5.5), the error energy of the reconstructed signal is upper bounded by

$$\|\tilde{\mathbf{x}} - \mathbf{x}\|_2 \leq \frac{\|\boldsymbol{\eta}\|_2}{\cos(\theta_{\mathcal{S},\mathcal{F}})}, \tag{5.7}$$

where $\theta_{\mathcal{S},\mathcal{F}}$ is the maximum angle between the subspace of signals supported on $\mathcal{S}$ and the subspace of $\mathcal{F}$-bandlimited GS [116]:

$$\begin{array}{rl} \cos(\theta_{\mathcal{S},\mathcal{F}}) = & \inf_{\|\mathbf{z}\|=1} \|\boldsymbol{\Psi}_{\mathcal{S}}\mathbf{z}\|_2 \\ \text{subject to} & \mathbf{U}_{:,\mathcal{F}}\mathbf{U}^{\mathrm{T}}_{:,\mathcal{F}}\mathbf{z} = \mathbf{z} \end{array} \tag{5.8}$$

Therefore, the error bound (5.7) is minimized when the subspace of signals in $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$ supported on $\mathcal{S}$ are more aligned as possible. From (5.8), it is clear that $\cos(\theta_{\mathcal{S},\mathcal{F}}) = \mathrm{SV}_{\min}(\boldsymbol{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}})$; therefore, in order to minimize the upper bound of the reconstruction error in (5.7), the set $\mathcal{S}$ should maximize $\mathrm{SV}_{\min}(\boldsymbol{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}})$. This optimal sampling strategy will be further discussed in the next section.

5.1.2 Optimal Sampling Strategies

Due to irregular topology of graphs, not only the sample size can influence the reconstruction error when sampling and interpolating a GS by $\boldsymbol{\Psi}_{\mathcal{S}}$ and $\boldsymbol{\Phi}$, respectively, but also the nodes in $\mathcal{S}$ themselves. Therefore, whenever possible, it is favorable to select *where* to sample in addition to *how many* nodes. It is also worth remembering that not all set $\mathcal{S}$ with cardinality $|\mathcal{S}| = M \geq K$ is a uniqueness set for $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$. In the previous section, it was shown that the reconstruction ℓ_2 error of an approximately bandlimited GS is minimized when the chosen sampling set $\mathcal{S}$ maximizes $\mathrm{SV}_{\min}(\boldsymbol{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}})$. Therefore, the optimal sampling set $\mathcal{S}^{\mathrm{opt}}$ in the presence of model mismatching is:

$$\mathcal{S}^{\mathrm{opt}} = \arg\max_{\mathcal{S}\in\mathcal{A}_M} \ \mathrm{SV}_{\min}(\boldsymbol{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}}), \tag{5.9}$$

where $\mathcal{A}_M$ is the set of sampling sets $\mathcal{S}$ with $|\mathcal{S}| = M$ and satisfying the admissibility condition in Theorem 5.1.4.

Besides model mismatching, reconstruction errors can also arise due to measurement noise. Consider a zero mean and uncorrelated Gaussian vector $\boldsymbol{\eta}$ that is added to the sampled $\mathcal{F}$-bandlimited GS, $\mathbf{x}$, as follows:

$$\mathbf{x}_{\mathcal{S}} = \boldsymbol{\Psi}_{\mathcal{S}}\mathbf{x} + \boldsymbol{\eta}, \tag{5.10}$$

then the recovered signal $\tilde{\mathbf{x}}$ is

$$\tilde{\mathbf{x}} = \boldsymbol{\Phi}\mathbf{x}_{\mathcal{S}} = \boldsymbol{\Phi}\boldsymbol{\Psi}_{\mathcal{S}}\mathbf{x} + \boldsymbol{\Phi}\boldsymbol{\eta} = \mathbf{x} + \boldsymbol{\Phi}\boldsymbol{\eta}.$$

Since $\|\boldsymbol{\Phi}\boldsymbol{\eta}\|_2 = \|\mathbf{U}_{:,\mathcal{F}}\boldsymbol{\Sigma}\boldsymbol{\eta}\|_2 \leq \|\mathbf{U}_{:,\mathcal{F}}\|_2\|\boldsymbol{\Sigma}\|_2\|\boldsymbol{\eta}\|_2$ and $\|\mathbf{U}_{:,\mathcal{F}}\|_2$ and $\|\boldsymbol{\eta}\|_2$ are fixed, to minimize the error effect of $\|\boldsymbol{\Phi}\boldsymbol{\eta}\|_2$, it is necessary to minimize $\|\boldsymbol{\Sigma}\|_2$. Since $\boldsymbol{\Sigma}\boldsymbol{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}} = \mathbf{I}$, minimizing the largest singular value of $\boldsymbol{\Sigma}$, its spectral norm, is equivalent to maximizing the the smallest singular value of $\boldsymbol{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}}$, leading to the optimization problem in (5.9). This problem is equivalent to the E-optimal design problem from experimental design theory.

Finding the optimal set $\mathcal{S}^{\text{opt}}$ is a combinatorial optimization problem that can require an exhaustive search in all possible subsets of $\mathcal{V}$ with size M. A suboptimal solution can be obtained by the greedy[2] search Algorithm 1. If a set function f (i.e.: $f(\mathcal{S}) = \text{SV}_{\min}(\mathbf{U}_{[\mathcal{S}+n]:,\mathcal{F}})$ in (5.9) is submodular, then a greedy algorithm such as Algorithm 1 provides near optimal solution. Nonetheless, the set function from Algorithm 1 is not submodular, and there is no theoretical guarantee that the resulting sampling set $\widetilde{\mathcal{S}}$ from Algorithm 1 is close to a solution $\mathcal{S}^{\text{opt}}$ from (5.9). There are other sampling strategies derived from experimental design that can also be used in this context. For instance, if the covariance matrix of the random vector $\boldsymbol{\eta}$ in (5.10) is known, the interpolation operator can take this knowledge into account. Other sampling criteria for GS, optimization designs and greedy algorithms are reviewed in [116].

A common concern in the extension of classical DSP methods to GSP is that, depending on the application, graphs can become very large and the GFT, a fundamental component of GSP, is computed globally. Therefore, the aforementioned sampling method, that requires the computation of at least K Laplacian eigenvectors, becomes prohibitive as the size of the graph increases. To deal with the computational cost problem, inspired by *compressed sensing*, a different sampling strategy based on random sampling was proposed in [117] and which is addressed

[2] A greedy algorithm searches a local optimal solution at each stage of the algorithm. For a general greedy algorithm, there is no guarantee of finding a global optimal solution or even near-optimal solution.

Algorithm 1 E-optimal greedy sampling algorithm [68]

```
    Input: $\mathbf{U}_{:,\mathcal{F}}$ and $M$, the size of the sampling set
    Output: a subset of nodes $\mathcal{S}$
procedure
while $|\mathcal{S}| < M$:
    $m \leftarrow \text{argmax}_n f(\mathcal{S}) = \text{SV}_{\min}(\mathbf{U}_{[\mathcal{S}+n]:,\mathcal{F}})$
    $\mathcal{S} \leftarrow \mathcal{S} + [m]$
end
    return $\mathcal{S}$.
```

in Appendix B.

5.2 Interpolation of Bandlimited GS

Definition 5.1.2 concerns only on the sparsity of the GS. Moreover, the previous sections of this chapter present a combined design for sampling and interpolation operators. In this section, we briefly introduce an interpolation approach, independent of the sampling method, based on a different definition for bandlimited signals that takes into account which frequencies are nonzero.

Definition 5.2.1. *(Bandlimited GS 2, [86]) The signal* $\mathbf{x}$ *is said to be* ξ*-band-limited if* $\hat{x}_k = 0$ *for all* k *such that* $\lambda_k > \xi$. *The space of* ξ*-band-limited signals on the graph* $\mathcal{G}$ *with GFT* $\mathbf{U}^{\mathrm{T}}$ *will be analogous to the Paley-Wiener space and denoted by* $\text{BL}_\xi(\mathbf{U})$.

Different from $\text{BL}_\mathcal{F}(\mathbf{U})$, $\text{BL}_\xi(\mathbf{U})$ takes the values of the Laplacian eigenvalues into account. Given an ξ-bandlimited GS, we aim to recover the full original GS $\mathbf{x}$ from few samples [27]. Theorem 5.2.3 provides a condition on a subset $\mathcal{S}$ to be a uniqueness set for subspace $\text{BL}_\xi(\mathbf{U})$. Note that $\xi_1 < \xi_2 \implies \text{BL}_{\xi_1}(\mathbf{U}) \subset \text{BL}_{\xi_2}(\mathbf{U})$ Before Theorem 5.2.3, we introduce the definition of a Λ-set.

Definition 5.2.2. *The subset* $\mathcal{S} \subset \mathcal{V}$ *is a* Λ*-set if*

$$\|\mathbf{x}\|_2 \leq \Lambda \|\mathbf{L}\mathbf{x}\|_2 \tag{5.11}$$

holds for all GS $\mathbf{x}$ *supported on* $\mathcal{S}$, $\Lambda > 0$.

Theorem 5.2.3. *([86]) The set* $\mathcal{S} \subset \mathcal{V}$ *is a uniqueness set for* $\text{BL}_\xi(\mathbf{U})$ *if and only if its complement* $\overline{\mathcal{S}}$ *is a* Λ*-set with* $0 < \xi < \frac{1}{\Lambda}$.

Demonstração. Let $\mathbf{x}, \mathbf{y} \in \text{BL}_\xi(\mathbf{U})$; it is enough to show that if $\mathbf{x}_\mathcal{S} = \mathbf{y}_\mathcal{S}$, then $\mathbf{y} = \mathbf{x}$. Note that if $\mathbf{x} \in \text{BL}_\xi(\mathbf{U})$, then $\mathbf{u}_k^{\mathrm{T}}\mathbf{x} = 0$ $\forall k$ such that $\lambda_k > \xi$

$$\|\mathbf{L}\mathbf{x}\|_2 = \|\mathbf{U}^{\mathrm{T}}\mathbf{\Lambda}\hat{\mathbf{x}}\| \leq \max_{\lambda_k < \xi} \lambda_k \|\hat{\mathbf{x}}\| \leq \xi \|\mathbf{x}\|. \tag{5.12}$$

Suppose $\mathbf{x}_{\mathcal{S}} = \mathbf{y}_{\mathcal{S}}$ but $\mathbf{y} \neq \mathbf{x}$, then $\mathbf{x} - \mathbf{y} \in \mathrm{BL}_{\xi}(\mathbf{U})$ is supported on $\overline{\mathbf{S}}$ and, since $\overline{\mathbf{S}}$ is a Λ-set and $\xi < \frac{1}{\Lambda}$:

$$\|\mathbf{x} - \mathbf{y}\| \leq \Lambda \|\mathbf{L}(\mathbf{x} - \mathbf{y})\| \leq \Lambda\xi \|\mathbf{x} - \mathbf{y}\| < \|\mathbf{x} - \mathbf{y}\|, \tag{5.13}$$

where the second inequality follows from (5.12), contradicting that $\mathbf{x} \neq \mathbf{y}$. □

The following Theorem proved in [27] provides a cutoff frequency ξ^* for GSs supported on $\mathcal{S}$, that is, the maximum ξ such that any signal in $\mathrm{BL}_{\xi}(\mathbf{U})$ can be perfectly recovered given a known subset of samples $\mathcal{S}$.

Theorem 5.2.4. *Given a subset $\mathcal{S}$, let $(\mathbf{L}^2)_{\overline{\mathcal{S}}}$ denote the submatrix of $\mathbf{L}^2$ containing only rows and columns corresponding to nodes in $\overline{\mathcal{S}}$. Then the set $\mathcal{S}$ is a uniqueness set for $\mathrm{BL}_{\xi}(\mathbf{U})$ with $0 < \xi \leq \xi^* = \overline{\lambda}_{\min}$, where $\overline{\lambda}^2_{\min}$ is the smallest absolute eigenvalue of $(\mathbf{L}^2)_{\overline{\mathcal{S}}}$.*

Demonstração. Let $\mathbf{z}$ be a GS supported on $\overline{\mathcal{S}}$, then

$$\frac{\|\mathbf{L}\mathbf{z}\|_2^2}{\|\mathbf{z}\|_2^2} = \frac{\mathbf{z}_{\overline{\mathcal{S}}}^{\mathrm{T}}(\mathbf{L}^2)_{\overline{\mathcal{S}}}\mathbf{z}_{\overline{\mathcal{S}}}}{\|\mathbf{z}_{\overline{\mathcal{S}}}\|_2^2} = R(\mathbf{z}_{\overline{\mathcal{S}}}, (\mathbf{L}^2)_{\overline{\mathcal{S}}}) \tag{5.14}$$

is the Rayleigh quotient of $(\mathbf{L}^2)_{\overline{\mathcal{S}}}$. Since $\overline{\lambda}^2_{\min} = \min\limits_{\mathbf{z}_{\overline{\mathcal{S}}} \in \mathbb{R}^{N-M}} R(\mathbf{z}_{\overline{\mathcal{S}}}, (\mathbf{L}^2)_{\overline{\mathcal{S}}})$:

$$\frac{\|\mathbf{L}\mathbf{z}\|_2^2}{\|\mathbf{z}\|_2^2} \geq \overline{\lambda}^2_{\min} \iff \frac{1}{\overline{\lambda}_{\min}}\|\mathbf{L}\mathbf{z}\|_2^2 \geq \|\mathbf{z}\|_2^2, \tag{5.15}$$

which means that $\overline{S}$ is a $\frac{1}{\sqrt{\overline{\lambda}_{\min}}}$-set and by Theorem 5.2.4, $\mathcal{S}$ is a uniqueness set for $\mathrm{BL}_{\xi}(\mathbf{U})$ with $0 < \xi \leq \xi^* = \overline{\lambda}_{\min}$. □

On the guarantee of existence of a reconstruction formula for a sampled GS, [86, Theorem 4.1] states that, if $\mathcal{S}$ is a uniqueness set for $\mathrm{BL}_{\xi}(\mathbf{U})$, then there exists a frame $\{\mathbf{v}_m\}_{m \in \mathcal{S}} \subset \mathrm{BL}_{\xi}(\mathbf{U})$ such that

$$x_n = \sum_{m \in \mathcal{S}} x_m v_{mn}, \ \forall \mathbf{x} \in \mathrm{BL}_{\xi}(\mathbf{U}).$$

Therefore, once a cutoff frequency ξ^* is obtained, the reconstruction of any signal restricted to a uniqueness set is obtained by least square projection. Any bandlimited signal in $\mathrm{BL}_{\xi}(\mathbf{U})$, $\xi < \xi^*$, can be written as a linear combination of the Laplacian eigenvectors corresponding to eigenvalues smaller than ξ^*. Let us denote by $\mathcal{F}^*$ the index set of the Laplacian eigenvectors corresponding to eigenvalues smaller than ξ^*, then $\mathbf{x} = \mathbf{U}_{\mathcal{F}^*}\hat{\mathbf{x}}$ $\forall \mathbf{x} \in \mathrm{PW}_{\xi^*}(\mathbf{U})$. Because $\mathcal{S}$ is a uniqueness set for $\mathrm{BL}_{\xi}(\mathbf{U})$, it contains all the spectral information of any GS in $\mathbf{x} \in \mathrm{BL}_{\xi}(\mathbf{U})$ and then $\mathbf{x} = \mathbf{U}_{\mathcal{S}^*}\hat{\mathbf{x}}_{\mathcal{F}^*}$.

Therefore, given a subset of nodes $\mathcal{S}$, and $\overline{\mathcal{S}}$ the remaining unknown samples, let $(\mathbf{L}^2)_{\overline{\mathcal{S}}}$ be the submatrix of $\mathbf{L}^2$ with rows and columns corresponding to nodes in $\overline{\mathcal{S}}$, then $\mathcal{S}$ is a uniqueness set for $\mathrm{BL}_{\overline{\lambda}^*_{\mathrm{min}}}(\mathbf{U})$, where $\overline{\lambda}^2_{\mathrm{min}}$ is the smallest singular value of $(\mathbf{L}^2)_{\overline{\mathcal{S}}}$. $\mathrm{SV}^*_{\mathrm{min}}$ is the largest bandwidth such that a GS can be perfectly recovered from $\mathcal{S}$. The unknown GS supported on $\overline{\mathcal{S}}$ can be recovered by

$$\mathbf{x}_{\overline{\mathcal{S}}} = \mathbf{U}_{\overline{\mathcal{S}}\mathcal{F}^*}(\mathbf{U}^{\mathrm{T}}_{\mathcal{S},\mathcal{F}^*}\mathbf{U}_{\mathcal{S},\mathcal{F}^*})^{-1}\mathbf{U}^{\mathrm{T}}_{\mathcal{S},\mathcal{F}^*}\mathbf{x}_{\mathcal{S}}. \tag{5.16}$$

The interpolator (5.16) is equivalent to (5.5), except for the definition of the set $\mathcal{F}$.

5.3 Conclusion

This chapter presented sampling/interpolation strategies for bandlimited GSs. Two different definitions of bandlimited GSs were considered: the first one concerns the cardinality of the support of the GS in the spectral domain, whereas the second one concerns the cutoff frequency. In both cases, the interpolation is performed by least mean square. The chapter also considered the case of approximately bandlimited GSs and presented a sampling strategy, analogous to the E-optimal design, that minimizes the reconstruction mean-squared error.

The sampling method presented in Section 5.1 will be used in the application proposed in Chapter 8 in combination with DL to jointly interpolate and forecast time-varying GSs. Next chapter presents a brief review on DL models to forecast multivariate time series.

Capítulo 6

Deep Learning Applied to Forecasting Review

Due to advances in many positioning technologies such as remote sensors, mobile devices, drones, and the global positioning system (GPS), spatiotemporal (ST) data have been broadly generated. Mining and extracting knowledge from this kind of data can contribute to a variety of tasks in climate analysis, transportation management, internet of things, and other geographical phenomenon analysis. ST data can be seen as a network in which each of its elements is associated with a time series. Time varying networks can also be built upon data other than geographical, for example, the signal of fMRI measured across many brain regions.

One of the most common problems associated with ST data is forecasting. Classical predictive models generally assume independence of data samples and bypass relevant spatial information. Vector autoregressive (VAR), an statistical multivariate model, and ML approaches such as support vector regression (SVR) [45] and random forest regression [46] can achieve higher accuracy than the more traditional methods but still fail to capture spatial relation. Many artificial intelligence (AI) solutions rely on extracting the right features from a given set of data and feeding it to an appropriate ML algorithm. In speech recognition, for example, an estimation of the speaker's vocal tract provides a good idea about the gender of the speaker [51]. Nonetheless, in many tasks it is highly impractical to determine a good representation of data, for example, determining the pixels suitable to discriminate images of dogs and cats. Beyond mapping an input into an output, neural network (NN) have the capacity to learn useful representations of data that improve the mapping accuracy [51]. The growing of computational power, the advances in graphics processing units (GPU) and the large amount of available data in a highly variety of fields have empowered even more the use of DL on a wide range of technological research areas such as data mining, speech recognition, image analysis, computer vision , recommendation systems, and signal processing in general [118]. Other sciences have also benefited from DL: predicting the way molecules will interact with each other in order to help pharmaceutical companies design new drugs [119], predicting atomic forces in solid [120], modeling the brain [121], and return forecasting, risk modeling and portfolio construction in financial sciences [122].

This chapter introduces forecasting DL techniques that, combined with graph filtering and the sampling and interpolation methods presented in Chapter 5, will be used in the algorithm proposed in Chapter 8. This chapter also presents 3 graph convolutional neural networks (GCNs) that mark the intersection between DL and GSP. A good review on graph neural networks can be found in [87]. The prior background of neural networks and backpropagation required by this chapter can be found in [51, Chapter 6].

6.1 Forecasting with Neural Networks

Although both the input and output of a forecasting model can be a multivariate time series, consider, for simplicity, the problem of predicting the univariate value x_{t+1} based on its past values $x_{t-\tau+1}\,, x_{t-\tau+2}\,, \dots, x_t$, then each pair of data sample is $(\mathbf{x}^t, y^t)$ with $\mathbf{x}^t = [x_{t-\tau+1}\, x_{t-\tau+2}\, \dots\, x_t]^{\mathrm{T}}$ and $y^t = x_{t+1}$. A general fully connected (FC) NN architecture for this forecasting problem is represented in Figure 6.1. The *input layer* corresponds to the vector $\mathbf{x}^t$, the *output layer* corresponds to the output y^t, and the *hidden layers* correspond to the intermediate vectors of this learning model. Each hidden vector $\mathbf{h}_\ell = [h_{\ell 1}, h_{\ell 2}, ..., h_{\ell J_\ell}]^{\mathrm{T}}$ is obtained by:

$$\mathbf{h}_\ell = \alpha_\ell(\mathbf{W}_\ell \mathbf{h}_{\ell-1} + \mathbf{b}_\ell), \tag{6.1}$$

where the weight matrix $\mathbf{W}_\ell \in \mathbb{R}^{J_\ell \times J_{\ell-1}}$ and and the bias vector $\mathbf{b}_\ell \in \mathbb{R}^{J_\ell}$ are the learnable parameters of layer ℓ and α_ℓ is an activation function. In order to learn the appropriate weight matrices and bias vectors of the model, let $\mathcal{L}(\mathbf{y}, \tilde{\mathbf{y}})$ denote the loss function, then gradients are computed as

$$\frac{\partial \mathcal{L}(y,\tilde{y})}{\partial \mathbf{W}_\ell} = \frac{\partial \mathcal{L}(y,\tilde{y})}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial \mathbf{h}_\ell} \frac{\partial \mathbf{h}_\ell}{\partial \mathbf{z}_\ell} \frac{\partial \mathbf{z}_\ell}{\partial \mathbf{W}_\ell} \tag{6.2}$$

$$\frac{\partial \mathcal{L}(y,\tilde{y})}{\partial \mathbf{b}_\ell} = \frac{\partial \mathcal{L}(y,\tilde{y})}{\partial \tilde{y}} \frac{\partial \tilde{y}}{\partial \mathbf{h}_\ell} \frac{\partial \mathbf{h}_\ell}{\partial \mathbf{z}_\ell} \frac{\partial \mathbf{z}_\ell}{\partial \mathbf{b}_\ell} \tag{6.3}$$

with $\mathbf{z}_\ell = \mathbf{W}_\ell \mathbf{h}_{\ell-1} + \mathbf{b}_\ell$. The parameters are then updated at each iteration i by a gradient descent algorithm with learning rate η:

$$\mathbf{W}_\ell^{(i)} = \mathbf{W}_\ell^{(i-1)} - \eta \frac{\partial \mathcal{L}(y, \tilde{y}^{(i)})}{\partial \mathbf{W}_\ell} \tag{6.4}$$

$$\mathbf{b}_\ell^{(i)} = \mathbf{b}_\ell^{(i-1)} - \eta \frac{\partial \mathcal{L}(y, \tilde{y}^{(i)})}{\partial \mathbf{b}_\ell}. \tag{6.5}$$

Common loss functions $\mathcal{L}(\mathbf{y}, \tilde{\mathbf{y}})$ are the negative log likelihood $\mathcal{L}(\mathbf{y}, \tilde{\mathbf{y}}) = \sum_{t=1}^{T_\mathrm{b}} y^t \log \tilde{y}^t$ and the mean squared error (MSE) $\mathcal{L}(\mathbf{y}, \tilde{\mathbf{y}}) = \frac{1}{T_\mathrm{b}} \sum_{t=1}^{T_\mathrm{b}} (y^t - \tilde{y}^t)^2$, where T_b is the training batch size.

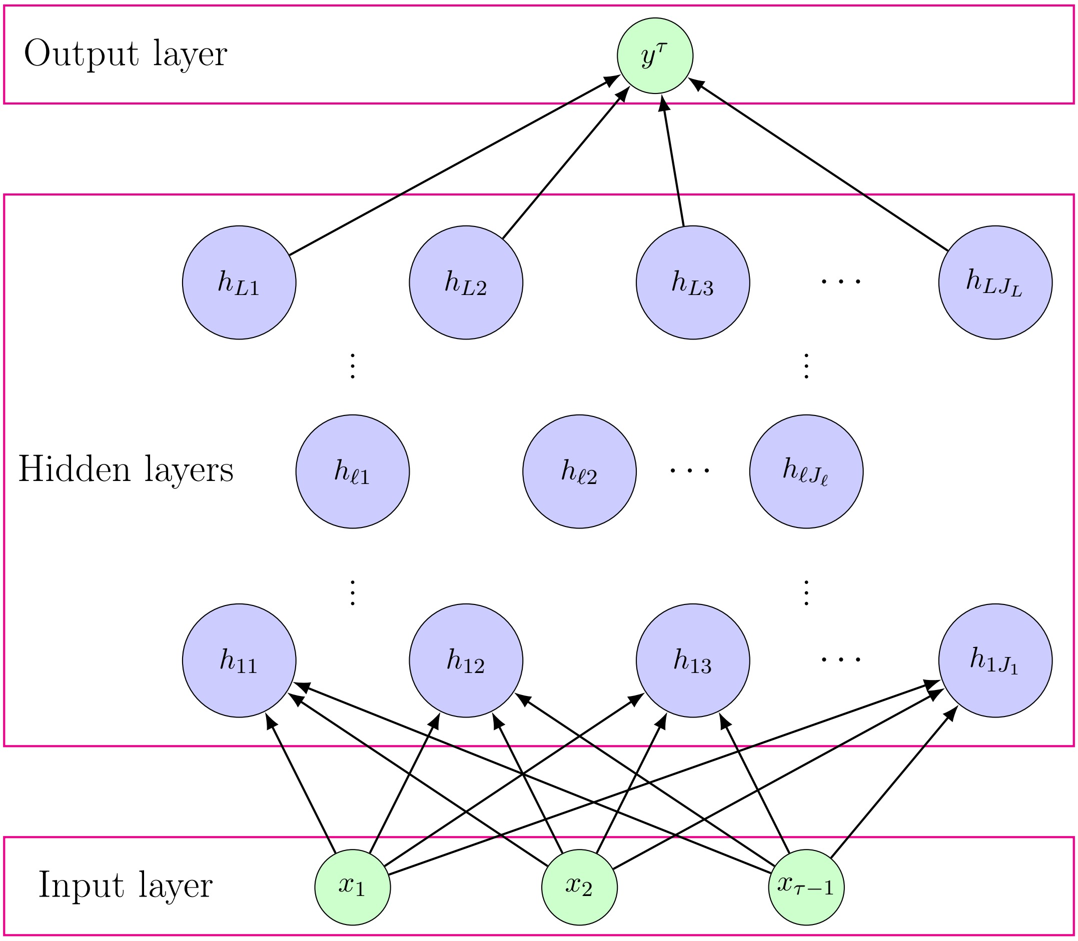


Figura 6.1: General NN architecture. This picture omits the bias added at each hidden layer

The architecture in Figure 6.1 has separate weight parameters for each time step, which impairs the generalization to sequences of variable lengths that are unseen in the training data. Moreover, consider the sentences “I am going to dance today” and “Today, I am going to dance”. Both sentences have the same meaning, although deliver the information in different ordering. To answer questions such as “When am

I going to dance?" or "What am I going to do?" based on the sentences above, a feed forward NN should learn all the meaning of words for each feature position (time position in the forecasting problem) separately. An RNN, on the other hand, avoids these problems by sharing parameters across feature positions [51, Chapter 10].

6.2 Recurrent Neural Networks

An RNN can be seen as a directed acyclic graph in which each neuron in the hidden layers represents a state. Figure 6.2 shows an RNN with a recursive connection, which loops τ times, on the left and the corresponding unfolded scheme with timestamp length $\tau = 3$ on the right. In a basic RNN with a single layer, for each timestamp $t \in \{1, 2, \ldots, \tau\}$, the hidden state $\mathbf{h}^t \in \mathbb{R}^{J_1}$ and the output $\mathbf{y}^t \in \mathbb{R}^{J_2}$ are computed as:

$$\mathbf{h}^t = \alpha_1(\mathbf{W}_{hx}\mathbf{x}^t + \mathbf{V}_{hh}\mathbf{h}^{t-1} + \mathbf{b}_1) \tag{6.6}$$

$$\tilde{\mathbf{y}}^t = \alpha_2(\mathbf{V}_{yh}\mathbf{h}^t + \mathbf{b}_2), \tag{6.7}$$

where α_1 and α_2 are activation functions, $\mathbf{W}_{hx} \in \mathbb{R}^{J_1 \times N}$, $\mathbf{V}_{hh} \in \mathbb{R}^{J_1 \times J_1}$, and $\mathbf{V}_{yh} \in \mathbb{R}^{J_2 \times J_1}$ are weight parameters and $\mathbf{b}_1 \in \mathbb{R}^{J_1}$ and $\mathbf{V}_{yh} \in \mathbb{R}^{J_2}$ are bias parameters. Note that the hidden state and the output do not necessarily have the same size as the input vector. Different from the previous section, we assume that both the input and the output are multivariate time series.

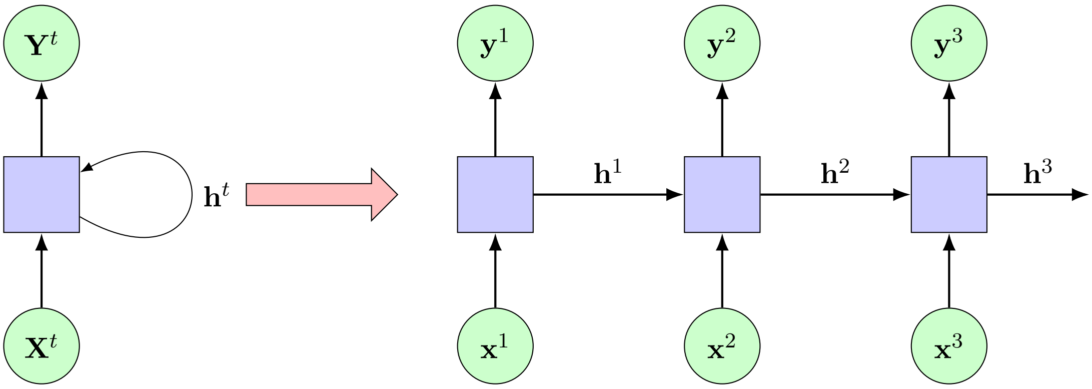


Figura 6.2: Basic RNN architecture. The vectors $\mathbf{X}^t = [\mathbf{x}^{t-\tau}\,\mathbf{x}^{t-\tau+1} \ldots \mathbf{x}^{t-1}]^{\mathrm{T}}$.

The network structure illustrated by Figure 6.2 is a *many-to-many* RNN, since both the input and the output have length lager than 1. Let $\mathbf{x}^1, ..., \mathbf{x}^{\tau_x}$ and $\mathbf{y}^1, ..., \mathbf{y}^{\tau_y}$ be the input output sequences of an RNN, respectively, then the RNN structure can be classified as:

- *one-to-one*: if $\tau_x = \tau_y = 1$;

- *many-to-one*: if $\tau_x > 1$ and $\tau_y = 1$;
- *one-to-many*: if $\tau_x = 1$ and $\tau_y > 1$;
- *many-to-many*: if $\tau_x > 1$ and $\tau_y > 1$ ($\tau_x = \tau_y$ or $\tau_x \neq \tau_y$).

This dissertation focus on the *many-to-one* structure and considerations on the other structures can be accessed in [51].

The main difference between RNNs and feedforward NNs is that an RNN can share parameters across the model. Sharing parameters is a key idea to deal with sequential data, since it allows the model to capture each piece of information regardless of where it occurs in the sequence [51, Chapter 10]. In Figure 6.2, each node has an output $\tilde{\mathbf{y}}^t$, which is used to compute the gradients of the learning model. Given a loss function $\mathcal{L}$ the gradients of the unfolded RNN with respect to W_{hx} and V_{yh} are:

$$\frac{\partial \mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)}{\partial \mathbf{W}_{hx}} = \frac{\partial \mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)}{\partial \tilde{\mathbf{y}}^t} \frac{\partial \tilde{\mathbf{y}}^t}{\partial \mathbf{h}^t} \frac{\partial \mathbf{h}^t}{\partial \mathbf{W}_{hx}}, \tag{6.8}$$

and

$$\frac{\partial \mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)}{\partial \mathbf{V}_{yh}} = \frac{\partial \mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)}{\partial \tilde{\mathbf{y}}^t} \frac{\partial \tilde{\mathbf{y}}^t}{\partial \mathbf{V}_{yh}}. \tag{6.9}$$

The gradient with respect to V_{hh}, following the product rule, is computed by differentiating each timestamp t and summing all together:

$$\frac{\partial \mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)}{\partial \mathbf{V}_{hh}} = \sum_{k=1}^{t} \frac{\partial \mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)}{\partial \tilde{\mathbf{y}}^t} \frac{\partial \tilde{\mathbf{y}}^t}{\partial \mathbf{h}^t} \frac{\partial \mathbf{h}^t}{\partial \mathbf{h}^k} \frac{\partial \mathbf{h}^k}{\partial \mathbf{V}_{hh}}. \tag{6.10}$$

Sharing parameters across timestamps allows the complexity of the model to not increase with the input length. Nonetheless, there are some drawbacks related to the recursive computation used by RNNs: computation can be slow for long historical data (large τ); failing to capture long range dependencies due to multiplicative gradients. This last one is also known as the *vanishing/explosion gradient* problem. By the chain rule in backpropagation through time (BPTT),[1] the gradient associated with weight parameters at the top of the RNN is a multiplicative chain of the derivatives in each state as illustrated by Figure 6.3. For example, unfolding the chain rule in equation (6.10) as:

[1] Backpropagtion through time is the optimization technique for training RNNs. It consists on a backpropagation algorithm applied to the unfolded RNN

$$\frac{\partial \mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)}{\partial \mathbf{V}_{hh}} = \sum_{k=1}^{t} \frac{\partial \mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)}{\partial \tilde{\mathbf{y}}^t} \frac{\partial \tilde{\mathbf{y}}^t}{\partial \mathbf{h}^t} \left(\prod_{j=k+1}^{t} \frac{\partial \mathbf{h}^j}{\partial \mathbf{h}^{j-1}} \right) \frac{\partial \mathbf{h}^k}{\partial \mathbf{V}_{hh}}, \tag{6.11}$$

then, for large t and small k, the term $\left(\prod_{j=k+1}^{t} \frac{\partial \mathbf{h}^j}{\partial \mathbf{h}^{j-1}} \right) = (\mathbf{V}_{hh})^{t-k} \prod_{j=k+1}^{t} \alpha'(\mathbf{z}^j)$, with $\mathbf{z}^j = \mathbf{W}_{hx}\mathbf{x}^j + \mathbf{V}_{hh}\mathbf{h}^{j-1} + \mathbf{b}^j$, can be a long multiplicative chain and may vanish of explode due to the term $(\mathbf{V}_{hh})^{t-k}$.

To deal with this problem, gating mechanisms have been used in the literature since they control the flow of information in the network. The two main gated RNNs, long short term memory (LSTM) [123] and gated recurrent unit (GRU) [124] will be presented in the following of this section. Besides RNNs, 1D convolution have been largely used to forecast multivariate time series and, for some datasets, may provide results on pair with RNNs with less computational time [125].

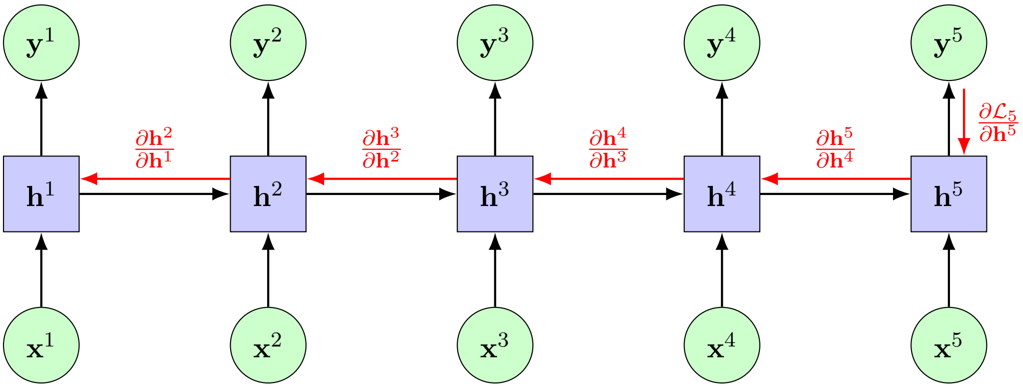

$\mathbf{y}^1$
$\mathbf{y}^2$
$\mathbf{y}^3$
$\mathbf{y}^4$
$\mathbf{y}^5$
$\frac{\partial \mathcal{L}_5}{\partial \mathbf{h}^5}$
$\frac{\partial \mathbf{h}^2}{\partial \mathbf{h}^1}$
$\frac{\partial \mathbf{h}^3}{\partial \mathbf{h}^2}$
$\frac{\partial \mathbf{h}^4}{\partial \mathbf{h}^3}$
$\frac{\partial \mathbf{h}^5}{\partial \mathbf{h}^4}$
$\mathbf{h}^1$
$\mathbf{h}^2$
$\mathbf{h}^3$
$\mathbf{h}^4$
$\mathbf{h}^5$
$\mathbf{x}^1$
$\mathbf{x}^2$
$\mathbf{x}^3$
$\mathbf{x}^4$
$\mathbf{x}^5$

Figura 6.3: Basic RNN BPTT.

6.2.1 Long Short Term Memory

An LSTM is an RNN in which each blue box in Figure 6.2 is replaced by the cell in Figure 6.4. The first step of an LSTM is the *forget gate layer* given by equation (6.12), that decides which information will be erased or not. The *forget gate* vector is $0 \preceq \mathbf{f}^t \preceq 1$, thus if f_n^t is close to zero, then the current state c_n^t is likely to be forgotten, if f_n^t is close to 1, on the other hand, then c_n^t is likely to be kept to the following recursion step.

The next step, the *input gate layer*, decides which new information will be stored in the current cell state $\mathbf{c}^t$. The equation corresponding to the *input gate* is shown by equation (6.13). Equation (6.14) shows the candidate $\tilde{\mathbf{c}}^t$ for the current state and equation (6.15) combines the gated previous current state and candidate state,$\mathbf{c}^{t-1}$ and $\tilde{\mathbf{c}}^t$, respectively, to update the new current state $\mathbf{c}^t$.

The last step of an LSTM, the *output gate layer*, represented by vector $\mathbf{o}^t$, regulates the information to be update to the hidden state $\mathbf{h}^t$:

$$\mathbf{f}^t = \sigma(\mathbf{W}_f\mathbf{x}^t + \mathbf{V}_f\mathbf{h}^{t-1} + \mathbf{b}_f) \tag{6.12}$$

$$\mathbf{i}^t = \sigma(\mathbf{W}_i\mathbf{x}^t + \mathbf{V}_i\mathbf{h}^{t-1} + \mathbf{b}_i) \tag{6.13}$$

$$\tilde{\mathbf{c}}^t = \tanh(\mathbf{W}_c\mathbf{x}^t + \mathbf{V}_c\mathbf{h}^{t-1} + \mathbf{b}_c) \tag{6.14}$$

$$\mathbf{c}^t = \mathbf{f}^t \odot \mathbf{c}^{t-1} + \mathbf{i}^t \odot \tilde{\mathbf{c}}^t \tag{6.15}$$

$$\mathbf{o}^t = \sigma(\mathbf{W}_o\mathbf{x}^t + \mathbf{V}_o\mathbf{h}^{t-1} + \mathbf{b}_o) \tag{6.16}$$

$$\mathbf{h}^t = \mathbf{o}^t \odot \tanh(\mathbf{c}^t) \tag{6.17}$$

where $\mathbf{f}^t, \mathbf{i}^t, \mathbf{o}^t \in \mathbb{R}^{J_1}$ are the vectors corresponding to the *forget gate*, *input gate*, and *output gate*, respectively; matrices $\mathbf{W} \in \mathbb{R}^{J_1 \times N}$ and $\mathbf{V} \in \mathbb{R}^{J_1 \times J_1}$ are the weight parameters; and $\mathbf{b} \in \mathbb{R}^{J_1}$ are the bias vectors.

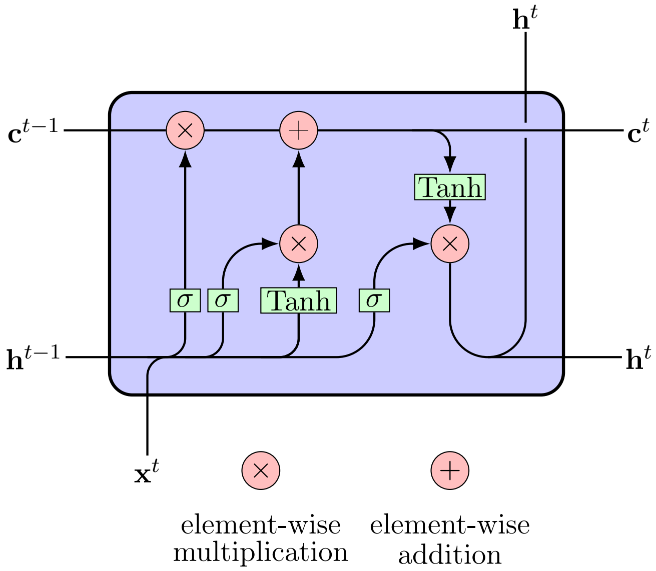


Figura 6.4: LSTM cell. Adapted from [126].

Although the BPTT of an LSTM depends on many little derivatives, we only present the derivatives necessary to understand how LSTMs deal with the *vanishing/explosion gradient* problem. In the BPTT of an LSTM, the gradients associated with $\mathbf{V} \in \{\mathbf{V}_f, \mathbf{V}_c, \mathbf{V}_i\}$ are computed similar to equation (6.11):

$$\frac{\partial \mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)}{\partial \mathbf{V}} = \sum_{k=1}^{t} \frac{\partial \mathcal{L}(\mathbf{y}^t, \tilde{\mathbf{y}}^t)}{\partial \tilde{\mathbf{y}}^t} \frac{\partial \tilde{\mathbf{y}}^t}{\partial \mathbf{h}^t} \frac{\partial \mathbf{h}^t}{\partial \mathbf{c}^t} \left(\prod_{j=k+1}^{t} \frac{\partial \mathbf{c}^j}{\partial \mathbf{c}^{j-1}} \right) \frac{\partial \mathbf{c}^k}{\partial \mathbf{V}}, \tag{6.18}$$

then by equation (6.15):

$$\begin{aligned}\frac{\partial \mathbf{c}^t}{\partial \mathbf{c}^{t-1}} &= \mathbf{c}^{t-1}\frac{\partial \mathbf{f}^t}{\partial \mathbf{c}^{t-1}} + \mathbf{f}^t + \tilde{\mathbf{c}}^t\frac{\partial \mathbf{i}^t}{\partial \mathbf{c}^{t-1}} + \mathbf{i}^t\frac{\partial \tilde{\mathbf{c}}^t}{\partial \mathbf{c}^{t-1}} \\ &= \mathbf{c}^{t-1}\frac{\partial \mathbf{f}^t}{\partial \mathbf{h}^{t-1}}\frac{\partial \mathbf{h}^{t-1}}{\partial \mathbf{c}^{t-1}} + \mathbf{f}^t + \tilde{\mathbf{c}}^t\frac{\partial \mathbf{i}^t}{\partial \mathbf{h}^{t-1}}\frac{\partial \mathbf{h}^{t-1}}{\partial \mathbf{c}^{t-1}} + \mathbf{i}^t\frac{\partial \tilde{\mathbf{c}}^t}{\partial \mathbf{h}^{t-1}}\frac{\partial \mathbf{h}^{t-1}}{\partial \mathbf{c}^{t-1}}. \end{aligned} \tag{6.19}$$

In order to compute the gradients in $\frac{\partial \mathbf{c}^t}{\partial \mathbf{c}^{t-1}}$, define $\Omega_\kappa^t = \mathbf{W}_\kappa \mathbf{x}^t + \mathbf{V}_\kappa \mathbf{h}^{t-1} + \mathbf{b}_\kappa$, $\kappa \in \{i, f, o, c\}$, then:

$$\begin{aligned}\frac{\partial \mathbf{c}^t}{\partial \mathbf{c}^{t-1}} =&(\mathbf{c}^{t-1} \odot \sigma'(\Omega_f^t) \odot \mathbf{V}_f)(\mathbf{o}^{t-1} \odot \tanh'(\mathbf{c}^{t-1})) + \mathbf{f}^t + \\ &(\tilde{\mathbf{c}}^t \odot \sigma'(\Omega_i^t) \odot \mathbf{V}_i)(\mathbf{o}^{t-1} \odot \tanh'(\mathbf{c}^{t-1})) + \\ &(\mathbf{i}^t \odot \sigma'(\Omega_c^t) \odot \mathbf{V}_c)(\mathbf{o}^{t-1} \odot \tanh'(\mathbf{c}^{t-1})),\end{aligned} \tag{6.20}$$

where $\sigma'(\Omega) = (1 - \sigma'(\Omega))^2$ and $\tanh'(\mathbf{c}^{t-1}) = 1 - \tanh^2(\mathbf{c}^{t-1})$. Note that, like (6.11), the derivative $\frac{\partial \mathbf{c}^t}{\partial \mathbf{c}^{t-1}}$ also depends on matrices $\mathbf{V}$ which could also lead to the problems of vanishing or explosion. Nonetheless, the gates in (6.20), that are set throughout the model's learning, may control the values of this derivative as convenient. For instance, suppose that $\frac{\partial \mathbf{c}^t}{\partial \mathbf{c}^{t-1}}$ starts to converge to zero, then if the training data presents long range dependencies, the forget gate $\mathbf{f}^t$ may take values close to 1 and avoid the gradient from vanishing [127].

6.2.2 Gated-Recurrent Unit

Different from the LSTM, the GRU cell is composed by only two gates $\mathbf{q}^t$ and $\mathbf{r}^t$, which modulate the flow of information inside the cell unit. Figure 6.5 depicts the architecture of a GRU. The gates are given by:

$$\mathbf{q}^t = \sigma(\mathbf{W}_q \mathbf{x}^t + \mathbf{V}_q \mathbf{h}^{t-1} + \mathbf{b}_q), \tag{6.21}$$

$$\mathbf{r}^t = \sigma(\mathbf{W}_r \mathbf{x}^t + \mathbf{V}_r \mathbf{h}^{t-1} + \mathbf{b}_r), \tag{6.22}$$

where $\{\mathbf{W}_q, \mathbf{W}_r\} \subset \mathbb{R}^{J_1 \times N}$ and $\{\mathbf{V}_q, \mathbf{V}_r\} \subset \mathbb{R}^{J_1 \times J_1}$ are matrices whose entries are learnable weights, $\{\mathbf{b}_q, \mathbf{b}_r\} \subset \mathbb{R}^{J_1}$ are the bias parameters.

The update of the hidden state $\mathbf{h}^t$ is a linear combination of the previous hidden state and the candidate state $\mathbf{c}^t$:

$$\mathbf{c}^t = \sigma(\mathbf{W}_c \mathbf{x}^t + \mathbf{V}_c(\mathbf{h}^{t-1} \odot \mathbf{r}^t) + \mathbf{b}_c), \tag{6.23}$$

$$\mathbf{h}^t = \mathbf{q}^t \odot \mathbf{c}^t + (1 - \mathbf{q}^t) \odot \mathbf{h}^{t-1}, \tag{6.24}$$

with $\odot$ being the element-wise multiplication. Similar to LSTM [128], the additive update of the hidden state can handle long-term dependencies by avoiding a quick vanishing of the errors in BPTT, and by not overwriting important features. The $\mathbf{r}^t$, called *reset gate*,controls how much of the previous information should be ignored before combining with $\mathbf{x}^t$ and $\mathbf{q}^t$, called *update gate*, controls how much of the past information will be kept to the next hidden state.

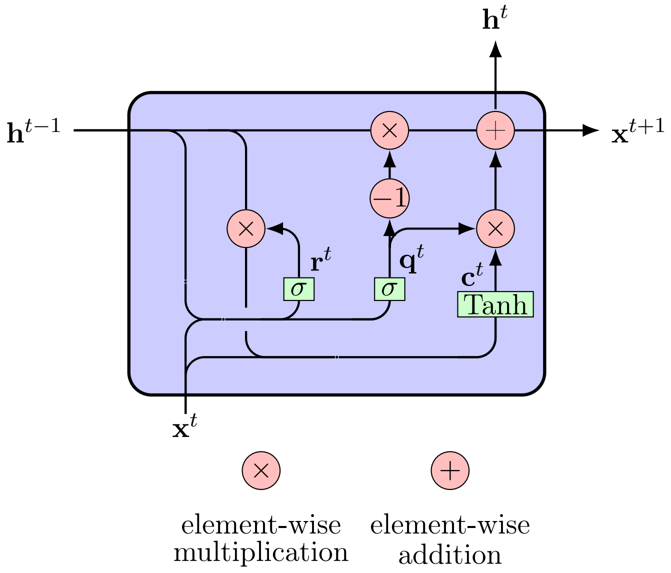


Figura 6.5: GRU cell.

6.2.3 RMSprop

Once the gradients are computed, the optimization of the parameters follows the same update rule from equation (6.4). Another common optimization algorithm broadly used to train models of sequential data is the RMSprop [129] which can be seen as an adaptation of the resilient backpropagation algorithm (Rprop) to deal with mini-batches. The Rprop uses only the sign of the gradient and adapt the step size of the update equation for each parameter separately. The RMSprop forces the normalization factor of the Rprop to be similar across adjacent mini-batches in order to combine the robustness of Rprop and the efficiency of min-batches. Let $\mathbb{E}(\nabla^i_{\mathbf{W}}\mathcal{L})^2$ denote the average of the squared gradients with respect to parameter $\mathbf{W}$ over a mini-batch at iteration i, then

$$\mathbb{E}[\nabla^i_{\mathbf{W}}\mathcal{L}]^2 = \beta\mathbb{E}[\nabla^{i-1}_{\mathbf{W}}\mathcal{L}]^2 + (1-\beta)(\nabla^i_{\mathbf{W}}\mathcal{L})^2 \tag{6.25}$$

$$\mathbf{W}^i = \mathbf{W}^{i-1} - \frac{\eta}{\sqrt{\mathbb{E}[\nabla^i_{\mathbf{W}}]}}\nabla^i_{\mathbf{W}}\mathcal{L} \tag{6.26}$$

The term $\mathbb{E}[\nabla^i_{\mathbf{W}}\mathcal{L}]^2$ is a moving average of the squared gradient and allows efficient learning from sequential data. The learning model proposed in Chapter 8 uses this algorithm for optimizing its parameters.

6.3 Graph Convolutional Neural Networks

Due to the widespread usage of graph-structured data in many applications, there is a growing interest in applying DL to this kind of data. Alongside RNNs, the convolutional neural network (CNN) is a successfully and broadly used class of NNs. The 2D-CNN is commonly used to capture spatial features but is restricted to grid-like uniformly structured data, such as images and videos. To extend CNNs to data defined on graphs, and inspired by GSP, some works have developed convolutions on GS [57–60, 130, 131]. This section presents three graph convolutional neural network (GCN) that have been adapted and combined with other methods in order to deal GS taking the underlying graph structure into account. The spectral graph convolution (SGC) proposed in [130] was the first attempt to extent CNNs to graphs using GSP. The spectral domain filters proposed by [130] are then replaced by polynomial approximation in [58] and a graph convolutional neural network (GCN) built in the vertex-domain based on the adjacency matrix is proposed by in [60].

CNNs are typically followed by global or local pooling layers, which reduce the dimension of the output after the convolution with the aim of reducing the complexity as the network become deeper. A local pooling, for example, uses statistical functions such as average and maximum value to aggregate small clusters in the 2D grid. In graphs, the notion of clusters is not straightforward then graph theoretical tools must be employed such as multi-resolution spectral clustering [132] used in [57]. In [58] the coarsening phase of the Graclus multilevel clustering algorithm [133] is used to coarsen the graph in each level of the network before pooling the GS. Moreover, in order to avoid unnecessary storage and improve the computational cost of the implementation, the nodes are sorted so that graph pooling becomes as efficient as conventional 1D pooling. Although pooling is a fundamental component of many deep NNs, it will not be covered by this dissertation.

6.3.1 Spectral Graph Convolution

A 2D-CNN consists of a 2D kernel, with width and height as hyperparameters, that slides through a 2-dimensional data. This convolution can be extended to 3 dimensional data with a $2D$ or a $3D$ kernel. The SGC replaces the 2D kernel by a graph filter and the convolution by the graph filtering in equation (2.10). The graph filter is built on the Laplacian matrix, although matrices based on the adjacency could be used as well. Each layer ℓ of the SGC transforms an input GS $\mathbf{x}_{\ell,i} \in \mathbb{R}^{N \times I_\ell}$ into an output GS $\mathbf{x}_{\ell,f} \in \mathbb{R}^{N \times I_\ell}$:

$$\tilde{\mathbf{x}}_{\ell+1,f} = \mathbf{U}\left(\sum_{i=1}^{I_{\ell-1}} \mathbf{W}_{\ell,f,i}\right)\mathbf{U}^{\mathrm{T}}\mathbf{x}_{\ell,i}, \tag{6.27}$$

$$\mathbf{x}_{\ell+1,f} = \alpha(\tilde{\mathbf{x}}_{\ell+1,f} + \mathbf{b}_{\ell,f,i}), \quad f \in \{1,\dots,I_\ell\}, i \in \{1,\dots,I_{\ell-1}\} \tag{6.28}$$

where $\mathbf{W}_{\ell,f,i} \in \mathbb{R}^{N\times N}$ is a diagonal matrix (graph filter in the spectral domain), $\mathbf{b}_{\ell,f,i}$ is the bias parameter ,α is an activation function, and I_ℓ is the number of filters in layer ℓ. Therefore, each layer ℓ has $N \times I_\ell \times I_{\ell-1}$ learnable parameters. Since smoothness is a common assumption for GSs, only the d first Laplacian eigenvalues are generally needed and the number of weight parameters becomes $d \times I_\ell \times I_{\ell-1}$. The choice of the cutoff frequency λ_d depends on the application as, in some cases, high frequencies may carry meaningful information.

To compute the gradients, denoting by $(w_{\ell,f,i})_k$ the k^{th} diagonal entry of $\mathbf{W}_{\ell,f,i}$, note that

$$(\tilde{x}_{\ell+1,f})_n = \sum_{k=1}^{d} U_{nk}(w_{\ell,f,i})_k \sum_{m=1}^{N} U_{mk}(x_{\ell,i})_m, \tag{6.29}$$

then, denoting by $\mathcal{L}$ the loss of the entire network,

$$\frac{\partial \mathcal{L}}{\partial (w_{\ell,f,i})_k} = \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_{\ell+1,f}} \frac{\partial \tilde{\mathbf{x}}_{\ell+1,f}}{(w_{\ell,f,i})_k} = (\bar{\mathbf{u}}_k^{\mathrm{T}}\mathbf{x}_{\ell,i})\mathbf{u}_k^{\mathrm{T}}\left[\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_{\ell+1,f}}\right] \tag{6.30}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_{\ell,f}} = \mathbf{U}\left(\sum_{i=1}^{I_{\ell-1}} \mathbf{W}_{\ell,f,i}\right)\mathbf{U}^{\mathrm{T}}\left[\frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_{\ell+1,f}}\right] \tag{6.31}$$

In [57] an algorithm to forward this network and compute gradients with complexity $O(1)$ is proposed. The computational drawback of this implementation is the multiplication by $\mathbf{U}^{\mathrm{T}}$ and $\mathbf{U}$ in each layer.

6.3.2 Spectral Graph Convolution in the Vertex Domain

In order to avoid the computation of the GFT, the graph convolution filters in [58] are approximated by the Chebyshev polynomials introduced by Section 2.2.5. The update equation of layer ℓ in (6.27) becomes:

$$\tilde{\mathbf{x}}_{\ell+1,f} = \sum_{i=1}^{I_{\ell-1}} \sum_{q=0}^{Q} c_{q,\ell,f,i} C_q(\mathbf{L})\mathbf{x}_{\ell,i}, \quad f \in \{1,\dots,I_\ell\}, i \in \{1,\dots,I_{\ell-1}\}, \tag{6.32}$$

where the Chebyshev coefficients $c_{q,\ell,f,i}$ are the parameters to be learned. Therefore, for each layer, there are $(Q+1) \times I_\ell \times I_{\ell-1}$ parameters to learn and gradients are

$$\frac{\partial \mathcal{L}}{\partial \mathbf{c}_{\ell,f,i}} = [\bar{\mathbf{x}}_{0,\ell,i}\, \bar{\mathbf{x}}_{1,\ell,i}\, \dots\, \bar{\mathbf{x}}_{Q,\ell,i}]^{\mathrm{T}} \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_{\ell+1,f}} \tag{6.33}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{x}_{\ell,i}} = \sum_{f=1}^{I_\ell} \left(\sum_{q=0}^{Q} c_{q,\ell,f,i} C_q(\mathbf{L}) \right) \frac{\partial \mathcal{L}}{\partial \tilde{\mathbf{x}}_{\ell+1,f}}, \tag{6.34}$$

where $\bar{\mathbf{x}}_{q,\ell,i} = C_q(\mathbf{L})\mathbf{x}_{\ell,i}$.

6.3.3 Low Order Approximation GCN

In the previous convolution neural network, the number of parameters per layer depends on the order of the approximation of the Chebyshev polynomial. In [60], low order approximation is proposed. Consider the normalized Laplacian matrix $\mathbf{L}_{\text{norm}} = \mathbf{D}^{-1/2}\mathbf{L}\mathbf{D}^{-1/2}$, then the largest eigenvalue of $\mathbf{L}_{\text{norm}}$ is bounded by 2 and $\overline{\mathbf{L}}$ in equation (2.15) becomes $\overline{\mathbf{L}}_{\text{norm}} \approx \mathbf{L}_{\text{norm}} - \mathbf{I}$, then approximating (6.32) with degree $Q = 1$ without the bias term is:

$$\begin{aligned} \tilde{\mathbf{x}}_{\ell+1,f} &= \sum_{i=1}^{I_{\ell-1}} c_{1,\ell,f,i}(\mathbf{L}_{\text{norm}} - \mathbf{I})\mathbf{x}_{\ell,i} + c_{0,\ell,f,i}\mathbf{x}_{\ell,i} \\ &= \sum_{i=1}^{I_{\ell-1}} -c_{1,\ell,f,i}\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}\mathbf{x}_{\ell,i} + c_{0,\ell,f,i}\mathbf{x}_{\ell,i}, \end{aligned} \tag{6.35}$$

The number of learnable weights can be reduced to $c_{\ell,f,i} = c_{0,\ell,f,i} = c_{1,\ell,f,i}$ and the layer equation (6.35) becomes $\sum_{i=1}^{I_{\ell-1}} c_{\ell,f,i}(\mathbf{I}+\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2})\mathbf{x}_{\ell,i}$. In order to have more stability of the GCN, the term $\mathbf{I}+\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$ is normalized as follows:

$$\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I} \text{ and } \tilde{D}_{nn} = \sum_{m=1}^{N} \tilde{A}_{nm},$$

then graph convolutional layer ℓ can be expressed as

$$\tilde{\mathbf{X}}_{\ell+1} = \tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{A}}\tilde{\mathbf{D}}^{-1/2}\mathbf{X}_\ell\mathbf{C}_\ell, \tag{6.36}$$

where $\mathbf{X}_\ell \in \mathbb{R}^{N \times I_{\ell-1}}$ is the input matrix and $\mathbf{C}_\ell \in \mathbb{R}^{I_{\ell-1} \times I_\ell}$ is the matrix of weight parameters from layer ℓ.

6.3.4 Implementation Aspects

Due to the recent increasing interest in the application of learning algorithms to graph structured data, some useful libraries have been developed. The *PythorchGeometric* [134] is a library in PyTorch implementing a diverse range of DL methods on graphs and irregular structures in general. The library provides tools for dealing with giant graphs and some benchmark datasets. The *deep graph library* [135] is also a DL library developed to deal with data residing on graphs. This library is agnostic to the DL framework and provides a back-end adapter interface. In [136], an overview of 120 datasets, available at [137], of varying sizes from a wide range of applications is analyzed.

6.4 Conclusion

Part I of this dissertation presented the fundamentals of GSP as well as the VFA and sampling/interpolation strategies for bandlimited or approximately bandlimited GSs. Part II employs the presented GSP tools to deal with time-varying GS in two different applications: the SGWT, in combination with machine learning algorithms, will be used in the task of anomaly detection of individual graph nodes; and the sampling/interpolation methods, in combination with the forecasting-DL approaches reviewed in this chapter, will be used to simultaneous forecast and interpolate GSs.

Parte II

Applications

Capítulo 7

Anomaly Detection in Graph Signals

Anomaly detection is the identification of instances of data differing significantly from the majority of the dataset and is applied in a variety of problems, such as fraud and intrusion detection, medical imaging, and event detection in sensor networks [138]. These different instances of data are also called outliers and are defined by Hawkings [139] as

> "An outlier is an observation which deviates so much from the other observations as to arouse suspicious that it was generated by a different mechanism."

It is important to point out that in many applications data may present a large amount of noise that may not be of any interest to the analyst. As noisy data can deviate significantly from the rest of data, it can sometimes be considered an outlier. Thus, in this chapter we will consider, as an outlier, a point that can be generated by a different process of the majority of data and an anomaly will only refer to the outlier that is interesting to the analyst. Detecting these anomalies may be extremely important and provides useful insights to many applications, including graph structured data. For example, an unusual behavior in computer networks can be generated by a malicious activity and its recognition is indispensable for the integrity of the system [32].

Common approaches to detect anomalies rely on defining a space for normal data and label as anomalies all the data points that lies out of this space. Although these methods seems simple, there are some difficulties: (i) the space of normal data can be hard to define and may not have precise boundaries, specially in the presence of noise. Moreover, the space of normal data may change over time so that algorithms need to be frequently adapted; (ii) if the anomaly is generated by a malicious mechanism, it will be always adapted to mimic the normal data, making even more difficult to define the normal space; (v) since anomalies are rare events, usually there are not enough anomalous data to develop some kind of algorithms;

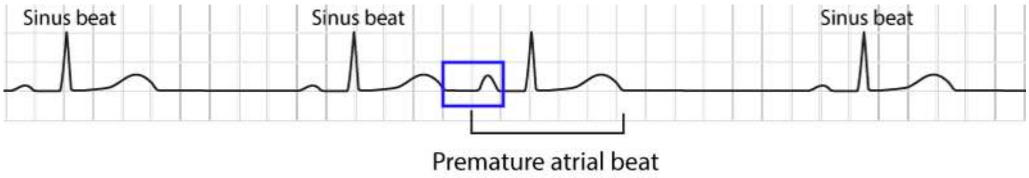


Figura 7.1: Collective anomaly: premature atrial contraction [140].

(iv) the notion of anomaly varies across different applications and algorithms may not be directly transferred from one application to another one [138].

Beyond these difficulties, there are some aspects of the data that should be taken into account when employing an anomaly detection algorithm, such as the nature of the data, the availability of labels, and the type of the anomaly to be detected. Data can be categorized in multiple ways, for example, data can be univariate, multivariate or multivariate with mixed types of variables. The types of variables are binary, discrete numerical, continuous numerical or categorical and different methods should be applied to deal with each of this kind of data. Moreover, there may exist some relation between instances of data (i.e.: sequential or spatial) requiring different treatment from the one used for independent data instances. Regarding the anomalies themselves, they can be categorized into 3 main classes:

1. Point anomaly: point anomalies are the most common anomalies and refer to data points that differs from the rest of the data. For example, consider a fraud card detection problem: if a client makes transactions of at most \$100.00 using a credit card, and suddenly appears a transaction of \$1000.00 in the credit card recordings, this transaction is highly likely to be a fraud (an point anomaly), since it totally deviates from the common behavior of the consumer. Thus, a anomaly detection algorithm would detect this potential fraud and report it to the client.

2. Collective anomaly: a collection of related data points deviates from the rest of the dataset is called collective anomaly. This type of anomaly is related to dependent data and, individually, each instance of data is not anomalous. T. A common example of collective anomaly is the premature atrial contraction detected in the electrocardiogram (ECG). Figure 7.1 illustrates a human ECG [140]. The blue box highlights a premature atrial contraction, that is, the heart P wave (small wave) corresponding the the atrial contraction occurred earlier so that the highlighted region is shorter than expected. The highlighted region is a collective anomaly because although each temporal instance has an acceptable value for an ECG, together they are abnormal.

3. Contextual anomaly: a contextual anomaly corresponds to data instances that are abnormal conditional to a given context. This type of anomaly is related

to data having two different information: the context (i.e: day of the year) and the behavior (i.e: average temperature in a day). For example, suppose it is 40°C in July at Rio de Janeiro, the behavior itself, 40°C, is not abnormal for Rio de Janeiro, but taking the context (winter), into account, this temperature can be considered anomalous.

The third aspect to be considered by an anomaly detection algorithm is the availability of data. As aforementioned, anomalous data are typically rare, impairing, for example, the use of supervised learning algorithms. The algorithms for anomaly detection can be (i) supervised, examples of both inliers[1] and abnormal points are available. These algorithms usually see the anomaly detection as a classification problem in which one class is composed by normal data whereas the other class is composed by anomalies; (ii) semi-supervised if only normal data or anomalous data are available. In this case, a common approach is to use the normal data points to define the normal space and points falling out of this region are labeled as anomalies. For example, a predictive learning model can be trained on the normal data and the label of anomaly can be assigned to data points whose observed values largely deviated from the prediction; (iii) unsupervised if there is no labeled data. A great review on anomaly detection algorithms can be found in [138].

Anomalies can also occur in data residing on graphs. This chapter focus on contextual anomalies in which the GS is the behavior and the underlying graph is the context. The proposed framework applies VFA concepts to the problem of anomaly detection in graphs. We also consider each graph node as an instance of data and applications in which the entire GS is considered as a single data point will not be covered. The idea behind using VFA to detect abnormalities in GS is that the spectral pattern of the data may provide some information on the expected behavior of each graph node. Section 7.3 provides experimental results with real data.

In [33], the authors employs GSP to detect anomalies in a sensor network of weather stations in the United States. In the absence of malfunctioning sensors in the system, it is expected that stations close to each other have similar temperature measurements and, therefore, the temperature GS is supposed to be smooth ($\mathcal{F}$-bandlimited with $\mathcal{F}$ corresponding to the smallest frequencies). Based on this hypothesis, high frequency components of the signal are extracted by a high-pass filter. If the maximum absolute value of the Fourier coefficients of the resulting filtered signal exceeds a threshold, an anomaly is detected in the *whole network*. The threshold is chosen as the maximum GFT absolute value of the high-pass filtered signals in the previous three days. This approach detects malfunctioning in the network but does not provide any clue about *where* is the corrupted measurement.

[1] The inliers correspond to the common data points

7.1 VFA-based Anomaly Detection

Aligned with the work in [33], a VFA-based classification method to detect and localize point anomalies in GS is proposed by this chapter [141]:

(i) define the adjacency matrix $\mathbf{A}$ of the graph based on a similarity measure across nodes. Section 7.3 provides an example of a possible choice for $\mathbf{A}$ in sensor graphs. An important assumption to this approach is that connected nodes have similar measurements.

(ii) compute a VFA representation of the raw input data. For instance, SGWT provides RN coefficients, which can all be used as new features, or just a subset of scales (e.g., coefficients of a single specific scale $r \in \{1, ..., R\}$).

(iii) feed a machine-learning-based or statistics-based classifier with the new transformed features.

In the case of sensor graphs, features can be chosen as a range of previous measurements (i.e. previous 6 months or previous 3 days). If vertex n is corrupted, $x^g_{n,r}$ of the current feature is expected to be different from the previous transformed features $x^g_{n,r}$.

7.2 Outlier Detection Algorithms

The third step of the methodology described above can involve a machine-learning classifier. This section introduces three machine-learning algorithms commonly used for outlier detection: local outlier factor (LOF), isolation forest (IF) and one class support vector machine (OCSVM). These algorithms are implemented in the *scikit learn* library and can be used for both unsupervised outlier detection or semi-supervised detection, also called novelty detection.

7.2.1 Local Factor Outlier

LOF [142] is an unsupervised algorithm that provides a score of how likely a data point is to be anomalous based on its k-nearest neighbors. More precisely, it compares the density of data points around a certain point to the density of points around its nearest neighbors. First, consider the set of k-nearest neighbors from a point p denoted by $\mathcal{N}_{\bar{k}}(p)$ (do not confuse with the k-hop ball $\mathcal{N}_k(n)$ from definition 2.1.2, the purpose of this section is introducing some algorithm detection algorithms and is not directly related to GSP) and the k-distance $d_k(p)$ that is the distance d (i.e. Euclidean distance) from point p to its k^{th} nearest neighbor. Figure 7.2 shows an example of a 3-distance, $d_{\bar{3}}(A)$, based on the Euclidean distance. Note that point B

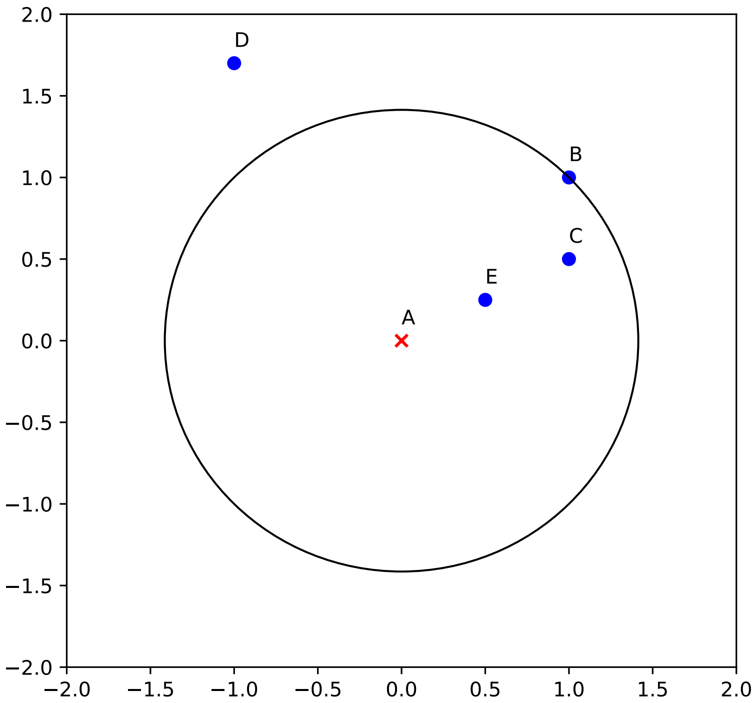


Figura 7.2: A 3-distance example.

is the third nearest point to A, then points B, C, and E are in $\mathcal{N}_{\bar{3}}(A)$ but point D is not. Second, define the *reachability distance* between two points p and q

$$\text{reach-dist}_k(p,q) = \max\left\{d_k(p), d(p,q)\right\}.$$

If point q is one of the k-nearest neighbors of p, then $\text{reach-dist}_k(p,q)$ is equal to the k-distance, otherwise it is the proper distance $d(p,q)$. Note that the reach-dist is not symmetric since point q can be in $\mathcal{N}_{\bar{k}}(p)$ but the converse may not hold. The reach-dist is used to introduce the concept of *local reachability density*, $\text{lrd}_k(p)$, which is the inverse of the average reach-dist value of points in $\mathcal{N}_{\bar{k}}(p)$:

$$\text{lrd}_k(p) = \frac{k}{\sum\limits_{q\in\mathcal{N}_{\bar{k}}(p)} \text{reach-dist}_k(q,p)}$$

The lrd of each point is then compared to the lrd of each of its k-nearest neighbors:

$$\text{lof}_k(p) = \frac{1}{k\text{lrd}_k(p)} \sum_{m\in\mathcal{N}_{\bar{k}}(p)} \text{lrd}_k(q).$$

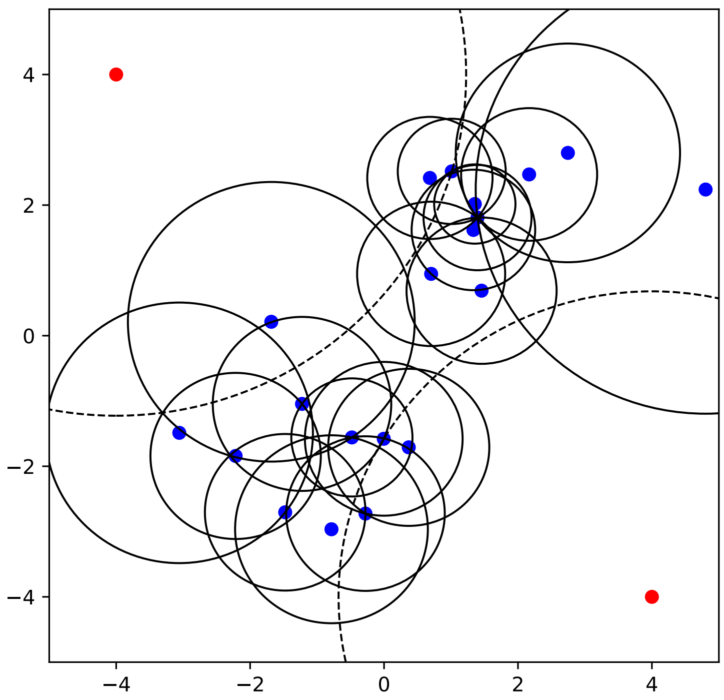


Figura 7.3: Comparison of 3-distance of normal data points and outliers.

If $\mathrm{lrd}_k(p) > 1$, point p is considered anomalous. In some applications, it is expected to have a given contamination rate $r_c \in [0, 0.5]$. In this case, a sample p is considered anomalous if $\mathrm{lrd}_k(p) < \mathrm{perc}_{r_c}(\mathrm{lof}_k)$ where $\mathrm{perc}_{r_c}(\mathrm{lof}_k)$ is the percentile r_c of the lof_k of all data points. Figure 7.3 shows the 3-distance of a dataset composed by 22 points. The red dots are outliers and have a large k-distance, represented by the circles centered at each data point, compared to common points, meaning that it is necessary to "walk" more through the data space until finding another data point. Dashed circles corresponds to the 3-distance of outliers. A disadvantage of LOF is that normal data points with not enough close neighbors may be misclassified. furthermore, the performance of the algorithm depends on the chosen distance d and have computational complexity $O(N^2)$.

7.2.2 Isolation Forest

This approach constructs an ensemble of tree structures to "isolate" each single instance of data. As anomalies are rare in a dataset and have values of features very different from those of common instances, anomalies are expected to be isolated closer to the root of the trees than normal instances of data [143]. An isolation tree (iTree) is a binary structure where each node has exactly zero or two daughter nodes, the last one is called test node. Each test node of the tree has a randomly selected feature f and a threshold t_f to separate data points into tow sets $x_f \leq f_t$ of $x_f > f_t$ until each tree node has exactly one instance of data. For each data point, the anomaly score of the iTree is taken as the path length (Section 2.1), denoted by $\ell(i)$, from the current leaf to the root of the tree, thus anomalies tends to have lower scores than normal data; .

In the IF, before the computation of each iTree, data is sampled and since the majority of data (normal) do not need to be separated, small sampling size is generally acceptable. The final score of data instance is taken as the average score for all the iTree's in the ensemble. In order to have comparable scores across the trees in the IF, the scores need to be normalized. Nonetheless, both the maximum and the average height of a iTree, grows as N and $\log N$, where N is the number of data instance. An approximation for the average path length, $\tilde{\ell}(i)$, was then borrowed from binary search three (BST) analysis:

$$\tilde{\ell}(i) = 2(\ln(i-1) + e) - 2\frac{i-1}{i}, \tag{7.1}$$

where e is the Euler's constant. Therefore, the outlier score of an instance n in a dataset with size N provided by an ensemble of N_{trees} is:

$$s(n, N) = 2^{-\frac{\sum_{m=1}^{N_{\text{trees}}} \ell_n(n)}{\tilde{\ell}(N)}}. \tag{7.2}$$

Then, if $\ell(n)$ is low on average, that is, point n is isolated near to the root of the iTree, then score $s(n, N)$ is close to 1 and n is classified as abnormal. If, on the other hand, $\ell(n)$ tends to $\tilde{\ell}(N)$, then $s(n, N)$ tends to 0.5 and n is classified as normal. When the percentage of contamination is provided, the data points are sorted according to the isolation score. Points at the top of the list (low score) are likely to be anomalies.

As the LOF, IF can also be employed to unsupervised and semi-supervised scenarios. Furthermore, IF takes some advantages: (i) it does not depend on computing any distance function between each pair of data points, having complexity $O(N)$; (ii) it scales well with the number of instances of data; (iii) its performance may not be impaired by high dimensional data points if some of the features (dimensions) are not informative [143].

7.2.3 One Class SVM

The main problem of applying support vector machine (SVM) to anomaly detection is that a simple SVM is designed to separate two classes of data by a decision boundary. In the anomaly detection problem, data is not labeled and all points are assumed to belong to a unique class. The OCSVM is an SVM algorithm addressed to anomaly detection [144].

Consider a transformation $\Phi(.)^2$ associated with a kernel $\mathcal{K}(.,.)$

$$\mathcal{K}(\mathbf{x}_n, \mathbf{x}_m) = \Phi(\mathbf{x}_n)^{\mathrm{T}}\Phi(\mathbf{x}_m) \tag{7.3}$$

mapping the feature space to another representation. The decision boundary that separates normal points and anomalies is

$$\mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}_n) - b = 0, \tag{7.4}$$

where $\mathbf{w}$ and b are learnable parameters. The optimization problem should be formulated in a way that $\Phi(\mathbf{x}_n) = \mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}) - b$ is positive for most of the data. The strategy is, then, to maximize the margin that separates the feature space provided by the feature map Φ from the origin:

$$\begin{array}{cc} \min\limits_{\mathbf{w},b,\boldsymbol{\xi}} & \frac{1}{2}\|\mathbf{w}\|_2 + \frac{1}{cN}\sum\limits_{n=1}^{N} \xi_n - b \\ \text{subject to} & \mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}_n) \geq b - \xi_n, n = 1, ..., N. \end{array} \tag{7.5}$$

, where $\boldsymbol{\xi} \in \mathbb{R}_+^N$ is a vector of slack variables, N is the number of training data points, and $c \in (0, 1)$ can be seen as the balance weight between normal and anomaly points. If c is small, then the second term of the objective function is more penalized and training data points tends to be classified as inliers. the Using (7.3), the decision function can be expressed in terms of the solution of the dual problem $\boldsymbol{\alpha} \in \mathbb{R}^N$ and the support vectors:

$$\operatorname{sign}\left\{\mathbf{w}^{\mathrm{T}}\Phi(\mathbf{x}_n) - b\right\} = \operatorname{sign}\left\{\sum_{m=1}^{N} \alpha_m \mathcal{K}(\mathbf{x}_m, \mathbf{x}_n) - b\right\}. \tag{7.6}$$

7.3 Experiments

In this section, we describe a numerical example on the Brazilian temperature network [145] in order to illustrate the framework for anomaly detection proposed in Section 7.1. The example employs both supervised and unsupervised scenarios.

[2] Despite similar notation, this section is not related to the interpolation operator from Chapter 5.

7.3.1 Graph Description

Data from [145] provides monthly average temperatures recorded by 296 Brazilian weather stations during the period of 1961-1990. The adjacency matrix $\mathbf{A}$ is constructed as equation (2.20). Unlike [33], the difference in altitude h_{mn} is taken into account because it is strongly correlated with temperature. In addition, all entries of $\mathbf{A}$ are divided by the largest eingenvalue in order to provide more stable operations.

7.3.2 Representation Using SGWT

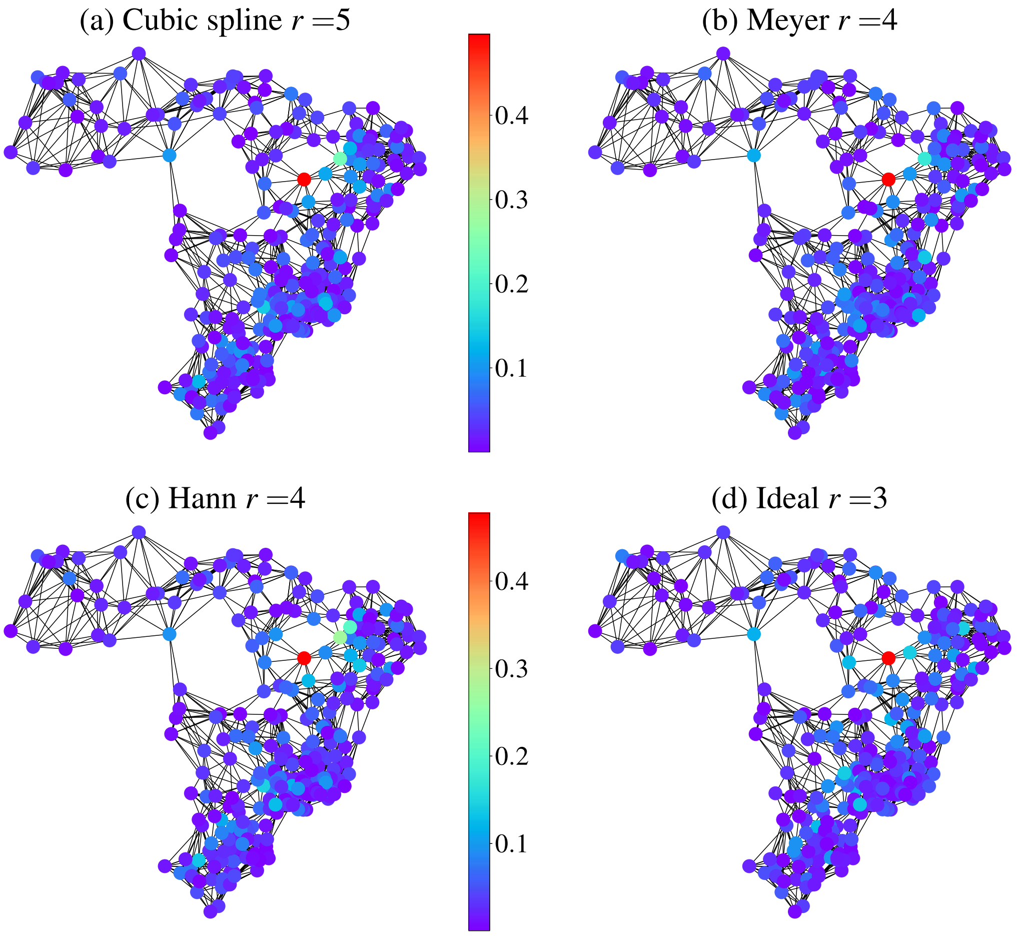


Figura 7.4: Magnitude difference between SGWT coefficients of non-corrupted and corrupted December temperatures for each wavelet kernel. Each r corresponds to the scale with the highest magnitude difference.

To illustrate the localization property of SGWT and its capability to detect deviating data, we use the actual temperature measurements of December for the GS $\mathbf{x}$. We also generated a disturbed GS by introducing an anomaly to a single vertex in the northeast region by adding 2°C, a mild drift, to its value. Each graph in Figure 7.4 depicts the energy per node of the absolute difference between the SGWT coefficients of the original signal, $x^g_{n,r}$, and the corrupted one, $\tilde{x}^g_{n,r}$, for a given scale r

and with n varying in $\{1, \ldots, N\}$. Wavelet atoms were constructed from the wavelet approximation in Figure 4.5. Each wavelet scale r is chosen such that the magnitude difference between coefficients of the corrupted node is the largest one. In fact, each SGWT detected a very localized variation in the transformed coefficients, but the largest magnitude differences appear in different scales of detail. For instance, the corruption of the vertex measurement was better detected by the scale of detail $r = 5$ when an SGWT with cubic spline kernel is applied, and by the scale $r = 3$ when an SGWT with ideal kernel is used.

7.3.3 SGWT-Based Anomaly Detector

Now, we showcase the methodology described in Section 7.1. We started from the original dataset in [145] to create a new dataset containing some malfunctioning weather stations. Then the usual division into training and test sets is considered to design and assess the performance of a VFA-based classifier.

To generate training anomaly/corrupted data, a Gaussian noise with variance $4^{\circ}\mathrm{C}^2$, truncated for absolute values smaller than 0.5, was added to the temperature in June in a quarter of the nodes, which were randomly selected. The set of non-corrupted training data was generated using another quarter of nodes, but now without modifying its corresponding temperature measurements. The test set was composed by the remaining nodes, and 50% of these nodes had their measurements corrupted, in December, by the same (truncated) Gaussian distribution as in the training set.

Following the general methodology in Section 7.2, the SGWT was applied to the six months before the current month (first semester for training and second semester for test) for each node in training and test sets in order to feed a Gaussian process classifier (GPC) [146] with radial basis function (RBF) kernel [147].[3]

For example, suppose node n is in the training dataset, then SGWT was applied to the signals of temperature in January, February, March, April, May and, June. The transformed coefficients $x^{g}_{n,r}$ for a predefined scale r were the extracted feature that fed GPC. The underlying hyperparameters are: kernel function, number of neighbors L in the adjacency matrix, number of wavelet scales R, wavelet coefficients used in feature extraction for a given scale r (we are using only one SGWT scale as features), and the order Q of the Chebyshev approximation. Table 7.1 contains the chosen hyperparameters for each kernel function and the respective $f1$-score.[4]

[3] A Gaussian process is collection of random variables in which each subset of the collection has a Gaussian distribution. GPC replaces the Gaussian prior on the weights of a logistic regression $\mathbf{w}$ by a Gaussian process prior in order to find $\mathbb{P}\{y = +1|\mathbf{x}, \mathbf{w}\} = \sigma_{\log}(\mathbf{x}^{\mathrm{T}}\mathbf{w})$, where $(\mathbf{x}, y)$ is the tuple of feature vector and respective label and $\sigma_{\log}$ is the logistic function.

[4] $f1$-score is a measure of accuracy that combines the precision p, which is the number of actual positive samples classified as positives divided by the total number of samples classified as positives,

Feeding GPC with raw vertex-domain data provided an average $f1$-score of 51%, much worse than using SGWT; the four VFA-based classifiers achieved an $f1$-score greater than 85% even with low-oder approximations.

Tabela 7.1: Hyperparameters of the VFA-based classifiers and corresponding $f1$-score

Kernel	L	R	r	Q	$f1$-score
Cubic spline	3	10	3	10	88%
Meyer	20	5	4	10	87%
Hann	20	5	3	3	86%
Ideal filter	10	5	5	3	85%

Unlike the scenario described in the previous experiment, where a balanced dataset was used, practical problems of anomaly detection have to deal with a small percentage of corrupted data. In order to evaluate the performance of the proposed VFA-based methodology from Section 7.1 in unbalanced scenario, a new dataset was generated for an unsupervised learning setup. This time, only 10% of the nodes from the entire dataset were corrupted by a Gaussian noise with variance $4^{\circ}C^{2}$ in the month of December. The SGWT was applied to the new dataset as in the supervised experiment and the transformed features fed the algorithms described in Section 7.2: LOF, IF and OCSVM with implementations available by the *scikit-learn* library. In order to report an average trend of the algorithm behavior without being biased by the particular choice of the corrupted nodes, the entire process (dataset definition and outlier detection) was repeated 50 times. Figure 7.5 depicts the *boxplot* of the overall performance achieved by each anomaly detection algorithm from Section 7.2 combined with the SGWT. The Evaluation metric is the receiver operating characteristic (ROC) curve, that measures true positive rate against false positive rate The boxplot shows the area under the ROC curve (AUC ROC). We also evaluated the score of the outliers detectors fed with raw data in vertex domain. Red lines represent the median of distributions, boxes represent the quartiles, and vertical lines extend to the most extreme non-outlier data point. Parameters in LOF and IF are default except for contamination (set as 0.2).

Most of the VFA-based classifiers outperformed the respective classifier fed with raw input, especially the LOF kernel. Low-order polynomial approximations of the graph filters were employed to achieve this good accuracy in detection. Although a low-order polynomial poorly approximates the Meyer SGWT, for example, it provides well-localized filters in the vertex domain (4.5), improving the specificity of the detector.

and recall r, which is the number of actual positive samples classified as positive divided by the number of actual positive samples. $f1$-score is then given by $2\frac{p.r}{p+r}$.

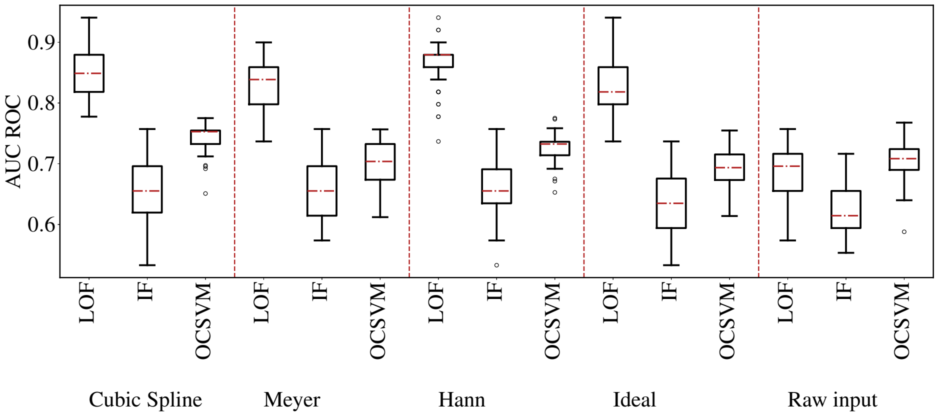


Figura 7.5: *Boxplot* of AUC of ROC for each combination of SGWT kernel and anomaly detection algorithms from Section 7.2.

An important concern in the unsupervised setting is the sensitivity to hyperparameters since a test set is not available for tuning. Figure 7.6 Shows the performance of these algorithms for different degrees of polynomial approximations of the wavelet kernels. The performance of the Cubic-spline-based classifier is more robust to the parameter Q whereas Meyer, Hann and the ideal kernels decrease the performance as Q increases. This is probably due to the bad localization in the vertex domain: for instance, for small Q, the Hann kernel got a better result when $R = 5$ and $r = 3$.

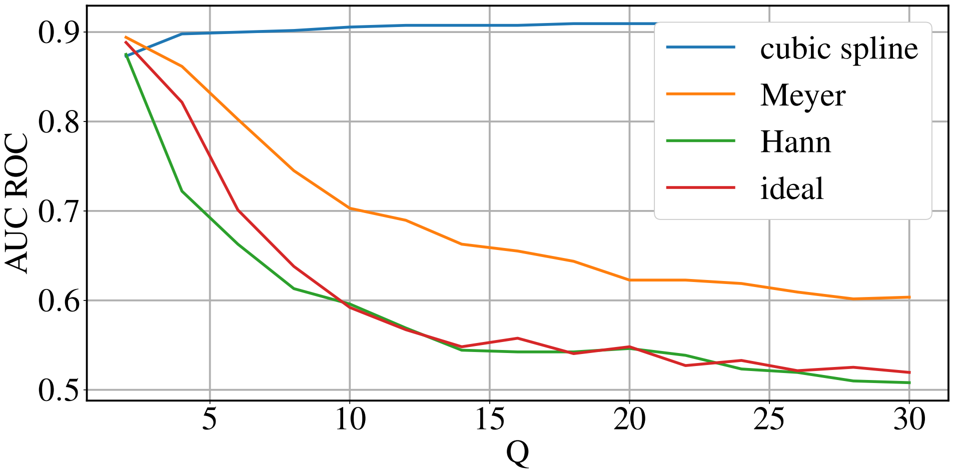


Figura 7.6: VFA-based classifiers performance for different approximation degrees. The other hyperparameters are set as in the previous experiments.

7.4 Final Remarks on the Results

This chapter employed the SGWT atoms to detect very small drifts of temperature in a sensor network, which is a type of contextual anomaly. The SGWT was used as feature extractor to feed a GPC, in a supervised experimental setup, or unsupervised anomaly detection algorithms, such as LOF, OCSVM, and IF. The results of this chapter showed that the VFA-based classifiers outperformed the classifiers fed with raw data. The low-order-polynomial approximations of the SGWT provided better results since their atoms are more localized in the vertex domain than the non-approximated atoms.

Capítulo 8

Joint Forecasting and Interpolation of GSs

This Chapter proposes a model for jointly forecasting and interpolating time-varying GSs. As introduced in Chapter 6, classical predictive models generally assume independence of data samples and disregard relevant spatial information [42], [43]. In the literature, In order to learn spatial information from multivariate time series, some works have combined CNNs [52–56] or the graph convolution NNs presented in Section 6.3 [57–60] with RNNs, such LSTM. These works are summarized in Table 8.1.

Two straightforward solutions to deal with the problem of forecasting and interpolating sampled GSs are:[1] (i) applying a forecasting model to the input GS and then interpolating the output; or (ii) interpolating the sampled GS and then feeding it to the forecasting model. These solutions tackle the ST prediction task separately and may fail to capture the inherent coupling between time and space domains. In this chapter, a graph-based NN architecture is proposed to handle ST correlations by employing GSP in conjunction with a gated-recurrent unit (GRU). Thus, we address the inherent nature of ST data by jointly forecasting and interpolating the underlying network signals. A global interpolation approach is adopted as it provides accurate results when the signal is smooth in the GSP sense, whereas an RNN forecasting model is adopted given its prior success in network prediction. Herein, not only the sampled GS is input to a predictive model but also its spectral components, which carry spatial information on the underlying graph. The major contribution of our proposed model is, therefore, the ability to learn ST features by observing a few nodes of the entire graph.

Considering the proposed learning framework, we introduce four possible classes of problems:

[1] We can regard a GS that will be interpolated as a sampled version of a GS defined over a denser set of nodes belonging to a virtual graph.

Tabela 8.1: Summary of recent works that use deep learning to predict ST data. First column defines the application. Second and third columns refer to the spatial and temporal techniques employed, respectively. "Conv." means temporal convolution; AE means auto encoder; RBM means restricted Boltzman machine; "other" encompasses other predictive strategies, such as attention mechanisms

Application	GSP	Temporal	Paper
traffic	Yes	RNN	[131], [148–153]
		Conv.	[154–159]
		other	[160], [161]
	No	RNN	[53], [56], [162–166],
		other	[38] ,[55], [167–171]
wind	Yes	RNN	[84]
	No	RNN	[50]
		Other	[172–174]
meteorological	Yes	AE	[175]
	No	RNN	[52]
		Other	[41, 176]
body-motion	Yes	RNN	[177–179]
neuroscience	Yes	Conv.	[180]
	No	RBM	[181]
semantic	Yes	RNN	[182]

- supervised applications, where the labels of all nodes are available for training but only a fixed subset of graph nodes can be used as input to the model in the test phase;

- semi-supervised application, wherein only data associated with a subset of nodes are available for training and computing gradients;

- noise-corrupted application, in which all nodes are available during the entire process, but additive noise corrupts the network signals;

- missing-value application, where a time-varying fraction of nodes are available for testing, but all nodes can be used for training.

The proposed approach achieves the best results in most tested scenarios related to the aforementioned applications, as compared to DL-based benchmarks.

The chapter is organized as follows: Section 8.1 describes the new learning framework. Section 8.2 describes four classes of applications that can benefit from the proposal. Section 8.3 presents the numerical results and related discussions.

8.1 Joint Forecasting and Interpolation of GSs

This Section proposes an ST neural network to jointly interpolate graph nodes and forecast future signal values. More specifically, the task is to predict the future state $\mathbf{x}^{t+p}$ of a network given the history $\mathbf{X}_{\mathcal{S}}^{t} = \{\mathbf{x}_{\mathcal{S}}^{t}, ..., \mathbf{x}_{\mathcal{S}}^{t-\tau+1}\}$.[2] Thus, the input signal is a GS composed by M nodes and the output GS is a network-signal snapshot composed by $N \geq M$ nodes. For now, to describe the learning model's architecture, we shall assume $p = 1$.

The proposed learning architecture employs a GRU cell [183] as the basic building block. The GRU structure, shown in Figure 6.5, was chosen to compose the forecasting module of the proposed method because its performance is usually on par with LSTM, shown in Figure 6.4, but with a lower computational burden [124]; nonetheless, it could be replaced by an LSTM or any other type of RNN or 1D convolutional layer.

8.1.1 Forecasting Module

The proposed learning model, named spectral graph GRU (SG-GRU), combines a standard GRU cell applied to the vertex-domain GSs comprising $\mathbf{X}_{\mathcal{S}}^{t}$ with a GRU cell applied to the frequency-domain versions of the latter GSs comprising $\hat{\mathbf{X}}_{\mathcal{F}}^{t}$. The GRU acting on frequency-domain signals is named here spectral GRU (SGRU), and has the same structure as the standard GRU, except for the dimension of weight matrices and bias vectors, which are $K \times K$ and K, respectively. The dimension of the hidden state in the SGRU is therefore K whereas the dimension of the hidden state in GRU applied to the vertex-domain is M.

Assuming that the entire GS$\mathbf{x}$ is $(\mathcal{F}, \epsilon)$-bandlimited, most of the information about it is expected to be stored in $\hat{\mathbf{x}}_{\mathcal{F}}$. Then, given an admissible operator $\mathbf{\Psi}_{\mathcal{S}}$, one has

$$\|\mathbf{x} - \mathbf{\Phi}_{\mathcal{S}}\mathbf{\Psi}_{\mathcal{S}}\mathbf{x}\|_2 \leq \frac{\epsilon}{\mathrm{SV}_{\min}(\mathbf{\Psi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}})}. \tag{8.1}$$

The choice of $\mathcal{F}$ will be further discussed in the experiments described in Section 8.3.

The SGRU module in the proposed learning framework is able to predict the (possibly time-varying) graph-frequency content of the network signals. This is key

[2] The GS at timestamp t is denoted by bold lowercase letter, $\mathbf{x}^{t}$, whereas the history set containing the sampled GSs in previous timestamps is denoted by bold capital letter, $\mathbf{X}_{\mathcal{S}}^{t}$.

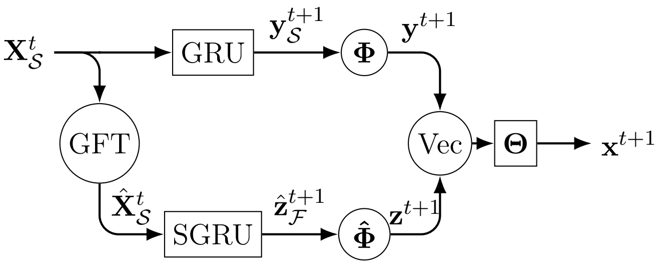

$\mathbf{X}_{\mathcal{S}}^{t}$
GRU
$\mathbf{y}_{\mathcal{S}}^{t+1}$
$\mathbf{\Phi}$
$\mathbf{y}^{t+1}$
GFT
Vec
$\mathbf{\Theta}$
$\mathbf{x}^{t+1}$
$\hat{\mathbf{X}}_{\mathcal{S}}^{t}$
SGRU
$\hat{\mathbf{z}}_{\mathcal{F}}^{t+1}$
$\hat{\mathbf{\Phi}}$
$\mathbf{z}^{t+1}$

Figura 8.1: Proposed SG-GRU model. The input GS follows two routes in parallel: in the upper route, the GRU followed by interpolation is applied to the GS; in the bottom route, the GS is transformed to frequency-domain before being processed by the SGRU module and thereafter being interpolated. The outputs of these two parallel processes are stacked into a single vector, represented by operation “Vec”, and fed to an FC layer.

to fully grasping the underlying spatial information embedded in the graph frequency content. Besides, it is worth pointing out that the proposed SGRU is slightly different from simply combining the SGC [130] described in Section 6.3 with a GRU: in both cases, the input signal is previously transformed to the Fourier domain, but in the SGRU a standard GRU, composed by matrix-vector multiplications, is applied to the transformed signal, whereas in the latter case, the SGC computes the graph convolution, which is an element-wise vector multiplication. Thus, the SGRU is able to better capture the temporal relations among different spectral components.

8.1.2 GS Interpolation

The outputs from the GRU, $\mathbf{y}_{\mathcal{S}}^{t+1}$, and from the SGRU ,$\hat{\mathbf{z}}_{\mathcal{F}}^{t+1}$, are interpolated by $\mathbf{\Phi}_{\mathcal{S}}$ and $\hat{\mathbf{\Phi}}_{\mathcal{S}} = \mathbf{\Phi}_{\mathcal{S}}\mathbf{U}_{:,\mathcal{F}}$, respectively. The resulting N-dimensional vectors $\mathbf{y}^{t+1}$ and $\mathbf{z}^{t+1}$ are stacked in a single vector of size $2N$ which is processed by a fully connected (FC) layer to yield

$$\tilde{\mathbf{x}}^{t+1} = \Theta(\mathbf{y}^{t+1}, \mathbf{z}^{t+1}), \tag{8.2}$$

as illustrated in Figure 8.1.

8.1.3 Loss Function

The loss function employed is the (empirical) mean square error (MSE). In a supervised scenario, the signal from all nodes are available for training, thus enabling the use of the entire GS$\mathbf{x}^{t+1}$ as label to compute the loss function. Given the batch size

T_b, the loss function for the supervised training $\mathcal{L}_\mathrm{s}$ is

$$\mathcal{L}_\mathrm{s} = \frac{1}{T_\mathrm{b}N}\sum_{t=1}^{T_\mathrm{b}} \|\tilde{\mathbf{x}}^{t+1} - \mathbf{x}^{t+1}\|_2^2. \tag{8.3}$$

In the semi-supervised task, on the other hand, only the sampled ground-truth signal $\mathbf{x}_\mathcal{S}^{t+1}$ can be accessed. In order to achieve better predictions on the unknown nodes, we propose to interpolate the sampled ground-truth signal by $\mathbf{\Phi}_\mathcal{S}$ before computing the MSE, yielding

$$\mathcal{L}_\mathrm{ss} = \frac{1}{T_\mathrm{b}N}\sum_{t=1}^{T_\mathrm{b}} \|\tilde{\mathbf{x}}^{t+1} - \mathcal{I}(\mathbf{x}_\mathcal{S}^{t+1})\|_2^2, \tag{8.4}$$

where

$$[\mathcal{I}(\mathbf{x}_\mathcal{S})]_n = \begin{cases} x_n, & \text{if } v_n \in \mathcal{S} \\ \left[\mathbf{\Phi}_\mathcal{S}\mathbf{x}_\mathcal{S}^{t+1}\right]_n, & \text{otherwise.} \end{cases} \tag{8.5}$$

8.1.4 Computational Complexity

The SG-GRU consists of two GRU cells, refereed as GRU and SGRU, which compute 6 matrix-vector multiplications each. The dimensions of the weight matrices in these recurrent modules applied on the vertex and frequency domains are M^2 and K^2, respectively, where K was set to $\frac{M}{3}$ in this paper (this choice will be further discussed in Section 8.3). The input of the SGRU is the sampled GS in the frequency domain, obtained by applying the truncated GFT, which is a $K \times M$ matrix. This transform can be pre-computed, avoiding the matrix vector multiplication during the loop recurrence. In this case the input of the network becomes a signal with dimension $M + K$. The output of the GRU and the SGRU are, thereafter, interpolated by $N \times M$ and $N \times K$ matrices, respectively, which are pre-computed before running the model. Finally, an FC layer is applied to the interpolated signals, costing $2N^2$ flops. Note that the truncated GFT, the interpolations, and the FC layers are out of the recurrence loop and do not increase the computational cost if a larger sequence length τ is used. Thus, the computational cost per iteration of the SG-GRU is

$$KM + 6\tau(M^2 + K^2) + N(K + M) + 2N^2 \text{ [flops]}. \tag{8.6}$$

8.2 Applications

The proposed learning architecture in Figure 8.1 can handle both supervised and semi-supervised scenarios. In the supervised case, measurements from the N

network nodes are available in the training step but not necessarily for testing. This supervised scenario covers many different applications; a case in point is a weather station network wherein the temperature sensors are working during a period of time, but then, suddenly, some of them are shut down due to malfunctioning or maintenance cost reduction. In the semi-supervised case, on the other hand, only part of the nodes appear in the training set and can, therefore, be used to compute gradients. Again, the semi-supervised scenario also covers many practical applications; for instance, when a sensor network is deployed with a limited number of nodes to reduce the related costs, but a finer spatial resolution is desirable, which can be obtained by a virtual denser sensor network.

Considering these two basic scenarios, we can conceive four specific types of applications:

8.2.1 Supervised Application

Input GS is composed by $M \leq N$ nodes but labels of all N nodes are used to compute the loss function in (8.3). As mentioned before, this learning model can be applied to situations in which all the N sensors are temporarily activated and, afterwards, $N - M$ sensors are turned off.

8.2.2 Semi-supervised Application

Both input GS and labels are composed by $M < N$. Thus, only the M in-sample are available to train the model using the loss function in (8.4). In this application, it is desired to predict the state of a static network with N nodes, considering that only $M < N$ sensors are deployed.

8.2.3 Noise-corrupted Application

Input GS is composed by all the N nodes with signals corrupted by uncorrelated additive noise, and the labels are the entire ground-truth GS. This application allows working with the proposed learning model when the sensors' measurements are not accurate. In this case, only the denoising capacity of the proposed model is evaluated, hence no sampling is performed over the input data.

8.2.4 Missing-value Application

Input GS is composed by all the N nodes but, at each time instant, a fraction of the N values measured by the sensor network are randomly chosen to be replaced by `NaN` (not a number). It is worth highlighting that this application is different from the (pure) supervised application in Section 8.2.1. In the supervised scenario, the set of known nodes, $\mathcal{S}$, is fixed across time, whereas the application of missing values considers different sets of known signal values at each time instant t. In other words, we have a supervised scenario with a time-dependent sampling set $\mathcal{S}^t$. The labels are the entire ground-truth GS. This setup evaluates the performance of the proposed SG-GRU when some of the sensors' measurements are missing, which could be due to transmission failures in a wireless network.

8.3 Numerical Experiments

In this section, we assess the performance of the proposed SG-GRU scheme in two real datasets. The simulation scenarios are instances of the four applications described in Section 8.2.

8.3.1 Dataset Description

The proposed learning model was evaluated on two distinct multivariate time-series datasets: temperatures provided by the Global Surface Summary of the Day Dataset (GSOD), which can be accessed at [184], and the Seattle Inductive Loop Detector Dataset (SeattleLoop) [162].

Global Surface Summary of the Day Dataset

The GSOD dataset consists in daily temperature measurements in °C from 2007 to 2013, totalling $2,557$ snapshots, in 430 weather stations distributed in the continental United States.[3] The source provides more weather stations but only 430 worked fully from 2007 until 2013. These stations are spatially represented by a 10-nearest-neighbor graph with nonzero edge weights given by:

$$A_{nm} = \frac{e^{-(d_{nm}^2 + h_{nm}^2)}}{\sqrt{\sum_{j\in\mathcal{N}_n} e^{-(d_{nj}^2+h_{nj}^2)}}\sqrt{\sum_{j\in\mathcal{N}_m} e^{-(d_{mj}^2+h_{mj}^2)}}}, \tag{8.7}$$

in which $\mathcal{N}_n$ is the set of neighboring nodes connected to the node indexed by n, whereas d_{nm} and h_{nm} are, respectively, the geodesic distance and the altitude difference between weather stations indexed by n and m. The adjacency matrix is symmetric and the diagonal elements are set to zero.

Seattle Inductive Loop Detector Dataset

The SeattleLoop dataset contains traffic-state data collected from inductive loop detectors deployed on four connected freeways in the Greater Seattle area. The 323

[3] Weather stations in the Alaska and in Hawai were not considered.

sensor stations measure the average speed, in miles/hour, during the entire year of 2015 in a 5-minute interval, providing $105,120$ timesteps. This dataset is thus much larger than GSOD. The graph adjacency matrix provided by the source [162] is binary and the GS snapshots are barely bandlimited with respect to the graph built on this adjacency matrix. To build a network model in which the SeattleLoop time series is $(\mathcal{F}, \epsilon)$-bandlimited with a reasonably small ϵ, the nonzero entries of the binary adjacency matrix were replaced by the radial-basis function

$$A_{nm} = \mathrm{e}^{-\frac{\|\mathbf{x}_n - \mathbf{x}_m\|^2}{10}}, \tag{8.8}$$

where $\mathbf{x}_n$ and $\mathbf{x}_m$ are time series, containing 1000 time-steps, corresponding to nodes v_n and v_m, respectively.

8.3.2 Choice of Frequency Set $\mathcal{F}$

The larger the set $\mathcal{F}$ the more information about the input signal is considered in the model. However, the interpolation using (5.5) is admissible only if $|\mathcal{F}| = K \leq M$ [68]. Moreover, if K increases, the smaller singular value of $\mathbf{U}_{\mathcal{S},\mathcal{F}}$ tends to decrease, leading to an unstable interpolation. Since the GSs considered in this paper are approximately bandlimited, using K close to M accumulates error during the training of the network. Based on validation loss, K was set to $\frac{M}{3}$.

When all nodes are available for training, that is, in the applications described in Sections 8.2.1, 8.2.3, and 8.2.4, $\mathcal{F}$ is chosen as the K Laplacian eigenvalues corresponding to the dominant frequency components (the ones with highest energy) of signals measured at the first 100 days. In the semi-supervised approach, on the other hand, the spectral content of the entire GS is unknown. Since the GSs considered in these experiments are usually smooth, in the sense that most of their frequency content is supported on the indices associated with the smaller Laplacian eigenvalues, the set $\mathcal{F}$ was chosen as the K smallest eigenvalues λ_n in this scenario. The set $\mathcal{F}$ used in the application described in Section 8.2.2 is, therefore, slightly different from the set $\mathcal{F}$ used in the scenarios related to the applications of Sections 8.2.1, 8.2.3, and 8.2.4.

8.3.3 Competing Learning Techniques

Recently many DL-based models were shown to outperform classical methods in the task of predicting ST data. Nonetheless, to the best of our knowledge, only [84] addresses the problem of predicting ST data by training a learning model with $M < N$ nodes, with the aim of reducing the training time duration. Therefore, the performance of our proposed method is here compared with DL-based models from the literature that do not actually handle sampled input GSs. Thus, we adapted the DL-based models from the literature by combining them with an interpolation strategy, such as k-NN and the GSP-based interpolator $\mathbf{\Phi}_{\mathcal{S}}$. In this context, the interpolation can be performed either: (i) before running the forecasting technique, so that the input of the competing DL-based model will be the entire GS; or (ii) after running the forecasting technique, so that the input of the competing DL-based model will be a sampled GS, thus requiring fewer learnable parameters.

We use as benchmark some LSTM-based NNs, which were shown to perform well in the strict forecasting task (i.e., time-domain prediction) on the SeattleLoop dataset in comparison with other baseline methods, such as ARIMA and SVR [153]. In addition, we also consider the spatiotemporal graph graph convolution network (STGCN) proposed in [154] as benchmark. In summary, the competing techniques (adapted to deal with sampled GSs) are:

(i) LSTM: simple LSTM cell;

(ii) C1D-LSTM: a 1D convolutional layer followed by an LSTM cell;

(iii) SGC-LSTM: the SGC from [130] (see Section 6.3.1)followed by an LSTM;

(iv) TGC-LSTM: a traffic graph convolution based on the adjacency matrix combined with LSTM [153];[4]

(v) STGCN: a combination of the graph convolution from [58] with a gated-temporal convolution [154]; Hyperparameters were set as in [154] since they lead to smaller MSE in the validation set (filter sizes were evaluated from the set $\{16, 32, 64\}$).[5]

As mentioned before, the above competing techniques do not tackle joint forecasting and interpolation tasks. Thus, they were combined with an interpolation technique. The output of methods (i)-(iv) were interpolated by $\mathbf{\Phi}_{\mathcal{S}}$, whereas a 1-hop neighborhood interpolation was applied before the method (v), that is, each unknown value x_n^t was set as

$$[x_n^t]_{\text{unknown}} = \frac{1}{|\mathcal{N}_n|} \sum_{m \in \mathcal{N}_n} x_m^t \,. \tag{8.9}$$

Unlike LSTM-based methods, the interpolation in (8.9) provided better results when combined with STGCN to handle the sampled input GS. The TGC-LSTM was only applied to the SeattleLoop dataset since it uses a free-flow reachability matrix, being specifically designed for traffic networks.

[4] Code from https://github.com/zhiyongc/Graph_Convolutional_LSTM .

[5] Code from https://github.com/VeritasYin/Project_Orion.

Table 8.2 summarizes the competing learning techniques along with the corresponding interpolation methods. “GSP interpolation” stands for interpolation by $\mathbf{\Phi}_{\mathcal{S}}$ and “1-hop interpolation” stands for interpolation by averaging records on the 1-hop neighborhood. Also the order followed by each procedure is indicated: “interpolation first” stands for interpolating the data before running the learning model, whereas “model first” refers to running the model and then interpolating its output.

Tabela 8.2: Summary of competing techniques

Model	**Interpolation**	**Order of procedures**
LSTM	GSP interpolation	model first
C1D-LSTM	GSP interpolation	model first
SGC-LSTM	GSP interpolation	model first
TGC-LSTM	GSP interpolation	model first
STGCN	1-hop interpolation	interpolation first

8.3.4 Figures of Merit

The prediction performance was evaluated by the root mean squared error (RMSE) and the mean absolute error (MAE):

$$\text{RMSE} = \frac{1}{T_{\text{t}}}\sum_{t=1}^{T_{\text{t}}}\left(\sqrt{\frac{1}{N}\sum_{n=1}^{N}(e_n^t)^2}\right), \tag{8.10}$$

$$\text{MAE} = \frac{1}{T_{\text{t}}}\sum_{t=1}^{T_{\text{t}}}\left(\frac{1}{N}\sum_{n=1}^{N}|e_n^t|\right), \tag{8.11}$$

where T_{t} is the number of test samples and e_n^t is the prediction error of the t^{th} test sample and n^{th} node. In the noisy setup, the mean absolute percentate error (MAPE) was also evaluated:

$$\text{MAPE} = \frac{1}{T_{\text{t}}}\sum_{t=1}^{T_{\text{t}}}\left(\frac{1}{N}\sum_{n=1}^{N}\frac{|e_n^t|}{|x_n^t|}\right) \times 100\%\,. \tag{8.12}$$

The error metrics MAE and RMSE have the same units as the data of interest, but RMSE is more sensitive to large errors, whereas MAE tends to treat more uniformly the prediction errors.

8.3.5 Experimental Setup

In the applications described in Sections 8.2.1 and 8.2.2, 75%, 50%, and 25% from the N nodes in $\mathcal{V}$ were selected to compose the set $\mathcal{S}$ using a greedy method of [185], called E-optimal design, with the set $\mathcal{F}$ corresponding to the first M smallest Laplacian eigenvalues. This choice of $\mathcal{F}$ relies on the smoothness of the underlying GS, that is, nodes near to each other are assigned with similar values. The same sampling sets were used for both supervised and semi-supervised training. All the experiments were conducted with a time window of length $\tau = 10$. The prediction length was $p = 1$ and $p = 3$ samples ahead for the GSOD dataset, that is, 1 day and 3 days, respectively, and $p = 1$ and $p = 6$ samples to SeattleLoop, that is 5 and 30 minutes, respectively.

The datasets were split into: 70% for training, 20% for validation, and 10% for test. Batch size was set to $T_{\mathrm{b}} = 40$ and the learning rate was 10^{-4}, with step decay rate of 0.5 after every 10 epochs. Training was stopped after 100 epochs or 5 non-improving validation loss epochs. The input of the model was normalized by the maximum value in the training set. The model was trained by the RMSprop [186] with PyTorch default parameters [129]. The network was implemented in PyTorch 1.4.0 and experiments were conducted on a single NVIDIA GeForce GTX 1080.

8.3.6 Results: Supervised Application

Table 8.3 and Table 8.4 show the MAE and RMSE in the supervised application. The proposed method outperformed all competitors in virtually all scenarios. When the sample size decreases, the performance gap increases compared to the benchmarks. On the GSOD dataset, the SG-GRU performed much better than the other strategies. We can see that, as the temperature GS is approximately ($\mathcal{F}$,ϵ)-bandlimited with small ϵ, the SG-GRU successfully captures spatial correlations by predicting the GSs' frequency content.

8.3.7 Results: Semi-supervised Application

The loss function in (8.4) was used for training the SG-GRU and the LSTM-based methods. For the STGCN, the interpolation of the target GS in (8.5) was replaced by the 1-hop interpolation. Table 8.5 and Table 8.6 show the result of the SG-GRU and the competing approaches on the SeattleLoop and GSOD datasets, respecti-

Tabela 8.3: MAE and RMSE of supervised prediction applied to the GSOD dataset

		$M = 0.75N$		$M = 0.50N$		$M = 0.25N$	
	Methods	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
p=1	**SG-GRU**	**1.66**	**2.21**	**1.73**	**2.28**	**1.74**	**2.31**
	LSTM	2.37	3.11	2.35	3.09	2.52	3.31
	C1D-LSTM	2.32	3.02	2.40	3.15	2.66	3.49
	SGC-LSTM	3.15	4.15	3.20	4.25	3.23	4.28
	STGCN	2.20	2.98	2.44	3.29	2.40	3.22

Tabela 8.4: MAE and RMSE of supervised prediction applied to the SeattleLoop dataset

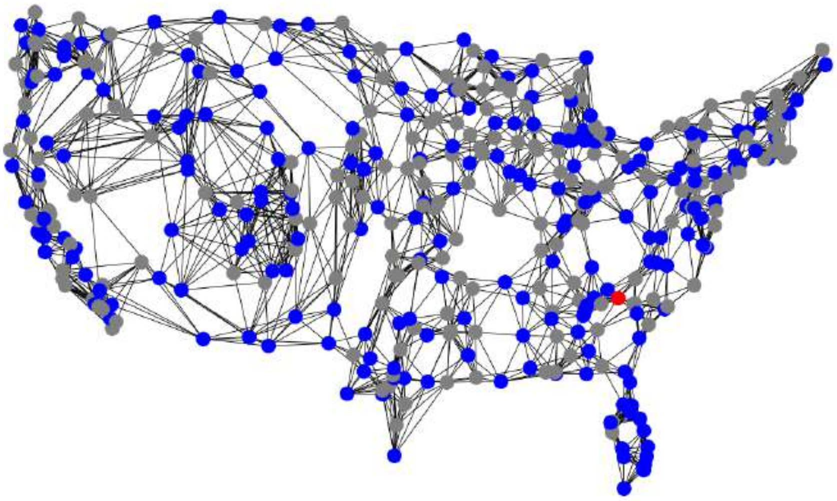
		$M = 0.75N$		$M = 0.50N$		$M = 0.25N$	
	Methods	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
p=1	**SGGRU**	**2.79**	**4.16**	**3.02**	**4.59**	**3.38**	**5.40**
	LSTM	3.15	4.79	3.64	5.59	4.45	7.03
	C1D-LSTM	3.25	4.95	3.70	5.70	4.49	7.08
	SGC-LSTM	3.59	5.57	3.97	6.14	4.60	7.26
	TGC-LSTM	3.03	4.59	3.54	5.45	4.40	6.98
	STGCN	**2.79**	4.32	3.11	4.82	3.65	6.10

vely. Figure 8.2b plots the outputs of the SG-GRU and LSTM methods, in the second semester of 2013 over the ground-truth signal, for a weather station out of the sampling set, highlighted in Figure 8.2a, considering a situation with 50% of known nodes. The SG-GRU outperformed the competing methods in the GSOD dataset. Since temperature GSs are highly smooth in the graph domain, the GSP interpolation, which is based on the assumption that the GS is bandlimited, provides good reconstruction. The energy of SeattleLoop dataset, on the other hand, is not as concentrated as the GSOD dataset, leading to a larger reconstruction error. Even with this limitation on the prior smoothness assumption, the SG-GRU outperformed the STGCN combined with 1-hop interpolation and the TGC-LSTM combined with GSP interpolation when the sampling set size is 25% or 50% of the total number of nodes. It is worth mentioning that the STGCN and the TGC-LSTM are learning models designed specifically for traffic forecasting. When the horizon of prediction is 30 minutes, then the SG-GRU achieved the smallest errors among all other methods. This could be due to simultaneous ST features extraction by the SGRU module. Figure 8.3 depicts the predicted speed by SG-GRU, STGCN, and TGC-LSTM for an unknown sensor with $p = 1$, $M = 0.50N$, and during the day 11/24/2015. As can be seen, the SG-GRU was able to better fit many points in the speed curve. It is worth mentioning that, despite the STGCN having poorly fitted

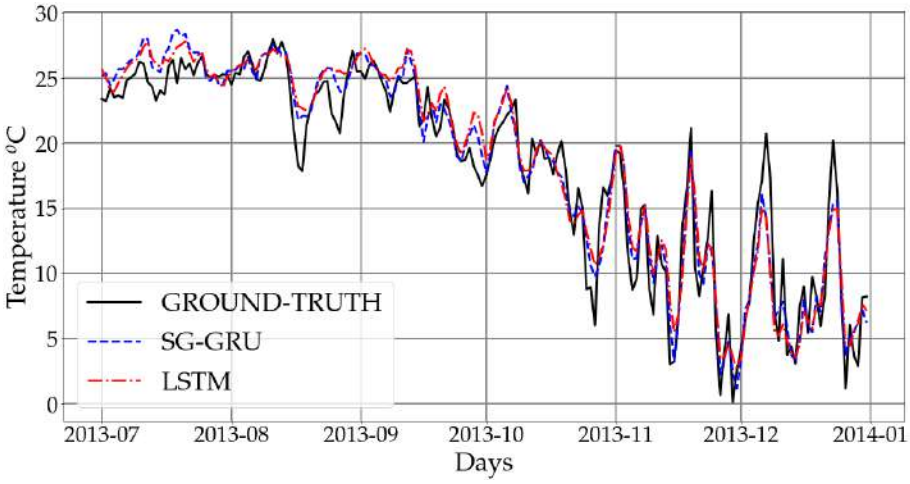
Tabela 8.5: MAE and RMSE of semi-supervised prediction applied to the GSOD dataset

		$M = 0.75N$		$M = 0.50N$		$M = 0.25N$	
	Methods	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
p=1	**SG-GRU**	**1.77**	**2.38**	**1.88**	**2.53**	**2.06**	**2.76**
	LSTM	2.35	3.03	2.41	3,16	2.72	3.54
	C1D-LSTM	1.83	2.44	2.00	2.65	2.24	2.97
	SGC-LSTM	2.75	3.66	2.84	3.76	3.01	3.97
	STGCN	2.34	3.2	3.75	5.02	6.92	8.65
p=3	**SG-GRU**	**2.84**	**3.76**	**2.90**	**3.85**	**2.99**	**3.94**
	LSTM	2.88	3.83	2.95	3.92	3.04	4.03
	C1D-LSTM	2.88	3.84	2.96	3.92	3.05	4.03
	SGC-LSTM	3.12	4.15	3.16	4.20	3.28	4.36
	STGCN	3.33	4.40	4.28	5.53	6.95	8.48

the curve in Figure 8.3b, it actually achieved higher accuracy on the known samples.

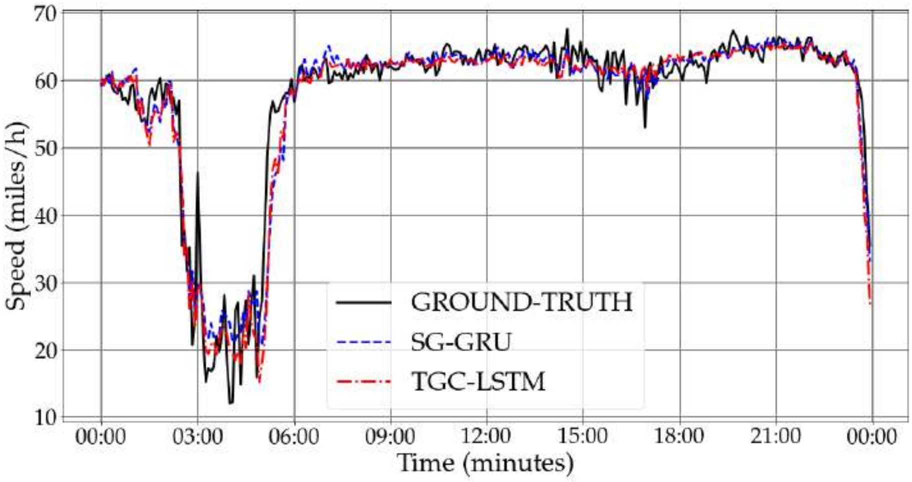


(a) US weather stations from GSOD dataset.

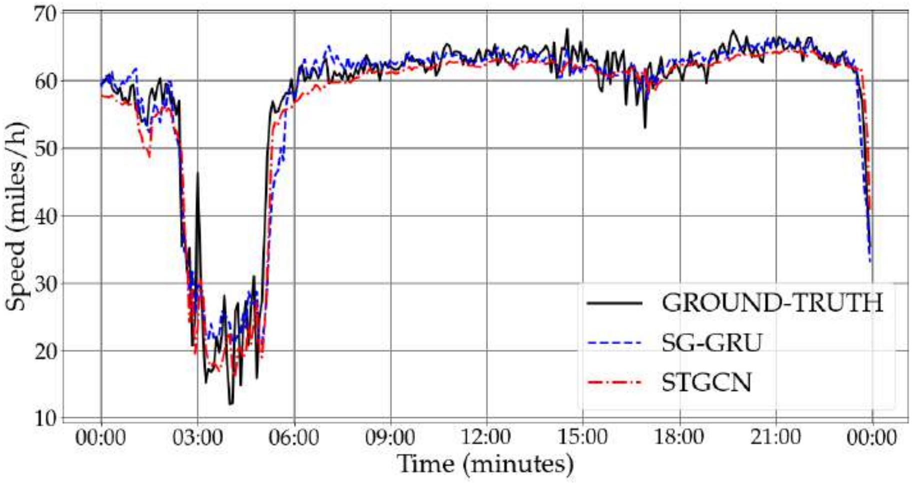


(b) Predicted temperature on a single sensor.

Figura 8.2: (a) Graph of sensors in the GSOD dataset. The known (50%) and unknown (50%) nodes are colored by blue and gray, respectively. The red node, which does belong to $\mathcal{S}$, indicates the weather station whose temperature predictions are shown in (b); and (b) output of the SG-GRU and the LSTM over the ground-truth temperature in the 2^{nd} semester of 2013 measured by the node highlighted in red in (a).



(a) SG-GRU and TGC-LSTM predictions.



(b) SG-GRU and STGCN predictions.

Figura 8.3: Predicted signal of the sensor i005es16920 using a subset with 50% of the nodes for the SG-GRU, TGC-LSTM, and STGCN. The evaluated sensor was absent in the sampling set $\mathcal{S}$.

8.3.8 Results: Noise-corrupted Application

In many real situations, sensors' measurements can be contaminated with noise, which may worsen forecasting accuracy. Therefore, to deal with these situations, it is important to develop robust algorithms. Consider a GS$\mathbf{x}$ with standard deviation σ_x and a measurement Gaussian noise, uncorrelated across both time and graph-domain, $\boldsymbol{\eta}$ with standard deviation σ_η. The noisy GS is $\tilde{\mathbf{x}} = \mathbf{x} + \boldsymbol{\eta}$ if the whole network is measured or $\tilde{\mathbf{x}}_{\mathcal{S}} = \boldsymbol{\Psi}_{\mathcal{S}}\mathbf{x} + \boldsymbol{\eta}$, if only the subset $\mathcal{S}$ is measured.

To evaluate the robustness of the proposed learning scheme, both SeattleLoop and GSOD datasets were corrupted by additive Gaussian noise with zero mean and standard deviation (std) $\sigma_\eta = 0.5\sigma_x$ and $\sigma_\eta = 0.1\sigma_x$, where σ_x is the std of the entire dataset: 10°C for GSOD dataset and 12.74 miles/h for SeattleLoop dataset. In this

Tabela 8.6: MAE and RMSE of semi-supervised approaches applied to the SeattleLoop dataset

		$M = 0.75N$		$M = 0.50N$		$M = 0.25N$	
	Methods	**MAE**	**RMSE**	**MAE**	**RMSE**	**MAE**	**RMSE**
p=1	**SG-GRU**	2.98	4.60	**3.53**	**5.55**	**4.50**	**7.28**
	LSTM	3.06	4.73	3.61	5.66	4.56	7.34
	C1D-LSTM	3.09	4.77	3.67	5.74	4.61	7.4
	SGC-LSTM	3.46	5.38	3.86	5.99	4.65	7.44
	TGC-LSTM	3.01	4.61	3.64	**5.55**	4.82	7.75
	STGCN	**2.88**	**4.35**	3.72	6.46	5.67	10.3
p=6	**SG-GRU**	**3.87**	**6.18**	**4.18**	**6.61**	**4.88**	**7.77**
	LSTM	3.96	6.34	4.31	6.81	4.98	7.94
	C1D-LSTM	3.96	6.37	4.3	6.83	5.02	7.97
	SGC-LSTM	4.12	6.63	4.44	6.98	5.03	7.98
	TGC-LSTM	4.91	7.89	5.17	8.23	8.29	12.5
	STGCN	4.54	6.82	4.56	7.90	6.11	10.8

experiment, nodes were not sampled and only the capability of handling noisy input was evaluated. Table 8.7 and Table 8.8 show MAE, RMSE, and MAPE[6] of the forecasting models respectively evaluated on 100 and 30 simulations of each of these noisy scenarios.

In the GSOD dataset, the proposed model achieved reasonable error levels in the presence of noisy measurements: for instance, MAE and RMSE increased 9% and 7% in comparison with the supervised situation with $M = 0.75N$ when the additive noise has std $\sigma_\eta = 0.1\sigma_x$. Many GS denoising approaches are based on attenuating high frequencies of the GS [187, 188]. The SGRU module of the proposed model promotes the smoothness of the predicted GS similarly: it runs a predictive algorithm over a restricted subset of the graph frequency content, $\mathcal{F}$, and thereafter computes the inverse GFT considering only this restricted subset,

In the SeattleLoop dataset, the MAE and RMSE evaluated on the proposed model increased 4% and 2%, respectively, in comparison with the supervised situation with $M = 0.75N$ when the additive noise has std $\sigma_\eta = 0.1\sigma_x$, This is a highly acceptable result, even though the STGCN achieved lower errors.

[6] Temperatures in the GSOD dataset were converted to Fahrenheit before computing MAPE to avoid division by zero.

Tabela 8.7: MAE, RMSE and MAPE (%) of forecasting applied to the GSOD with noise corruption

		$\sigma_\eta = 0.1\sigma_x$			$\sigma_\eta = 0.5\sigma_x$		
	Methods	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
p=1	**SG-GRU**	**1.81**	**2.36**	**7.70**	**2.01**	**2.61**	**8.52**
	LSTM	1.98	2.59	8.55	2.11	2.75	9.70
	C1D-LSTM	1.90	2.49	8.07	2.03	2.65	8.65
	SGC-LSTM	2.94	3.89	13.9	2.95	3.91	13.9
	STGCN	2.19	2.94	10.3	2.66	3.48	12.3
p=3	**SG-GRU**	**2.85**	**3.79**	**13.41**	**2.89**	**3.83**	**13.5**
	LSTM	2.88	3.83	13.6	2.93	3.88	13.8
	C1D-LSTM	2.86	3.8	13.4	2.91	3.85	13.6
	SGC-LSTM	3.18	4.21	15.2	3.16	4.2	15.2
	STGCN	3.17	4.23	15.6	3.21	4.25	15.7

Tabela 8.8: MAE, RMSE and MAPE (%) of forecasting applied to the SeattleLoop with noise corruption

		$\sigma_\eta = 0.1\sigma_x$			$\sigma_\eta = 0.5\sigma_x$		
	Methods	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
p=1	**SG-GRU**	2.91	4.27	13.7	3.13	4.66	16.1
	LSTM	3.21	4.85	18.2	3.45	5.20	19.4
	C1D-LSTM	3.30	5.05	19.4	3.48	5.30	20.5
	SGC-LSTM	3.96	6.28	28.3	4.07	6.44	29.9
	TGC-LSTM	2.88	4.24	13.0	3.19	4.74	14.2
	STGCN	**2.63**	**3.85**	**11.3**	**3.08**	**4.39**	**12.9**
p=6	**SG-GRU**	**3.74**	**6.02**	26.3	**3.84**	**6.19**	26.1
	LSTM	3.99	6.35	28.6	4.07	6.47	27.1
	C1D-LSTM	3.99	6.39	28.6	4.07	6.49	28.1
	SGC-LSTM	4.56	7.27	34.6	4.58	7.29	34.7
	TGC-LSTM	3.79	6.09	**26.1**	3.92	6.28	**25.3**
	STGCN	3.77	6.18	26.4	4.00	6.33	26.0

8.3.9 Results: Missing-value Application

Another common problem in real time-series datasets are missing values, which could occur due to sensor's malfunctioning or failure in transmission. To evaluate the performance of the SG-GRU in this situation, 10% of both SeattleLoop and GSOD datasets were randomly set to `NaN`. Before applying the forecasting methods, each NaN value, x_n^t, was interpolated by the 1-hop interpolation in (8.9). Table 8.9 and Table 8.10 show the numerical results of this scenario considering two forecasting horizons on the GSOD and SeattleLoop datasets, respectively. The forecasting accuracy decreases when there are missing values, as expected. For instance, in the GSOD dataset, MAE and RMSE increased 6% and 4% in comparison with the supervised situation with $M = 0.75N$. In the SeattleLoop dataset, MAE increases about 10% whereas the RMSE decreases about 8%. The GFT in the proposed model (and also in combination with the LSTM-based models) tends to smooth the output signal, reducing large deviations and consequently the RMSE. Nonetheless it can slightly increase the forecasting error across many nodes, leading to the increase in MAE.

Tabela 8.9: MAE, RMSE and MAPE (%) of forecasting applied to the GSOD dataset with 10% of missing values

	$p = 1$			$p = 3$		
Methods	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
SG-GRU	**1.75**	**2.3**	**7.53**	**2.87**	**3.77**	**13.1**
LSTM	2.56	3.41	12.3	2.94	3.93	14.1
C1D-LSTM	2.53	3.36	11.9	2.92	3.9	13.9
SGC-LSTM	3.51	4.81	20.2	3.22	4.34	16.8
STGCN	2.10	2.87	9.97	3.22	4.31	15.2

Tabela 8.10: MAE, RMSE and MAPE (%) of forecasting applied to the SeattleLoop dataset with 10% of missing values

	$p = 1$			$p = 6$		
Methods	**MAE**	**RMSE**	**MAPE**	**MAE**	**RMSE**	**MAPE**
SG-GRU	3.10	**3.85**	4.60	**6.17**	14.9	**23.7**
LSTM	3.37	4.07	5.08	6.49	19.6	27.3
C1D-LSTM	3.44	4.06	5.23	6.49	19.6	27.3
SGC-LSTM	4.01	4.58	6.34	7.31	16.3	35.1
TGC-LSTM	3.15	3.91	4.70	6.26	13.5	24.8
STGCN	**2.60**	3.91	**3.95**	6.26	**11.8**	24.9

8.3.10 Computational Cost and Efficiency

In the SeattleLoop Dataset, the epoch duration of SG-GRU was, on average, 8.5 s, whereas the more complex approaches, TGC-LSTM and STGCN, took around 40 s and 84 s per epoch, respectively. In the GSOD dataset, which is much shorter than the SeattleLoop, the average epoch duration of SG-GRU, LSTM, and STGCN were

0.20 s, 0.25 s, and 2.5 s, respectively. Table 8.11 shows the average training time, including pre-processing and data preparation, as well as test phases for the 3 semi-supervised scenarios applied on the SeattleLoop and GSOD datasets, with $p = 1$. The SG-GRU required more epochs to converge than STGCN, but it still trains faster than the STGCN and also than the other competing approaches.

Tabela 8.11: Average computational time in seconds

	SeattleLoop		GSOD	
Methods	**Training**	**Test**	**Training**	**Test**
SG-GRU	414.68	4.89	11.18	0.01
LSTM	1134.0	5.63	30.74	0.01
C1D-LSTM	1319.0	5.90	36.90	0.03
SGC-LSTM	2770.3	6.10	39.89	0.04
TGC-LSTM	1027.1	5.48	-	-
STGCN	725.58	12.6	83.92	0.12

8.3.11 Final Remarks on the Results

The consistently better results obtained by the SG-GRU for the GSOD dataset steem from the smoothness of the temperature GS with respect to the graph domain; SG-GRU relies on the assumption of bandlimited GSs. Therefore, SG-GRU is a promising approach to predict spatially smooth GSs. It is worth mentioning that the choice of the adjacency matrix is fundamental for a good performance, since it eventually defines the smoothness of the GSs. In the SeattleLoop dataset, which is not really smooth, the SG-GRU outperformed both the STGCN and the LSTM-based approaches when the sample size was small and the prediction time horizon was 30 minutes, thus indicating that the SG-GRU can capture ST dependencies by taking the network frequency content into account. Moreover, SG-GRU has low computational cost and can be boosted with more recurrent or fully connected layers, when sufficient computational resources are available.

Capítulo 9

Conclusion and Future Works

9.1 Concluding Remarks

This thesis presented the fundamental topics of GSP such as GFT, VFA, and graph sampling and tackled two different applications: VFA-based anomaly detection of time-varying GS and joint forecasting and interpolation of time-varying GS.

Part I presented a detailed review on the theory and techniques of GSP that have been developed in the last decade. In order to deal with graph-structured data, GSP extends the concepts and tools of DSP by considering the graph Laplacian or adjacency eigenvectors as the Fourier basis, thus the underlying structure of the graph is taken into account in the analysis/processing of the GSs. This thesis focused on VFA, analogous to the time-frequency analysis, which allows the spectral pattern of the signal to be accessed locally in the vertex-domain and is useful to analyze non-stationary GSs. Unlike classical DSP, the localization of VFA atoms varies across different nodes depending on the magnitude of the respective columns in the GFT matrix.

Other fundamental tools of signal processing addressed in this thesis are the operations of downsampling and upsampling. Unlike DSP, it is not obvious how to design these operators such that the downsampled GS is perfectly recovered. The solution adopted by many authors in the literature assumes the GS to be bandlimted, as in the Shannon-Nyquist sampling theorem. The original GS can be perfectly reconstructed by least mean square provided condition (5.3) is satisfied.

9.1.1 VFA-based Anomaly Detection

VFA is a useful tool to analyze GS simultaneously in frequency and vertex domains. In the anomaly detection experiment proposed in Chapter 7, the SGWT atoms were used to extract features from data, so that very small drifts in temperature measurements due to malfunctioning sensors could be detected with high accuracy by GPC, in a supervised experimental setup, and by unsupervised anomaly detection algorithms, having a clear advantage over classifiers that consider only the time-series structure of temperature measurements (like feeding GPC with raw data). Moreover, the low-order polynomial approximations of the SGWT provided better results then the high-order polynomial approximations by favoring localization in the vertex domain. Thus, VFA-based framework from Section 7.1 can be useful to the problem of detecting contextual anomalies, in which the context is represented by the graph.

9.1.2 Joint Forecasting and interpolation of GS

Chapter 8 presented a deep learning model for jointly forecasting and interpolating network signals represented by graph signals. The proposed scheme embeds GSP tools in its basic learning-from-data unit (SG-GRU cell), thus merging model-based and deep learning approaches in a successful manner. Indeed, the proposal is able to capture spatiotemporal correlations when the input signal comprises just a small sample of the entire network. Additionally, the technique allows reliable predictions when input data is noisy or some values are missing by enforcing smoothness on the output signals.

9.1.3 Future Works

There are three directions of interest for future works : (i) reducing the number of hyperparameters of the VFA-based framework proposed in Section 7.1; (ii) employing VFA to a semi-supervised anomaly detection setup; (iii) and studying theoretical arguments for the choice of the spectral set $\mathcal{F}$ in SG-GRU.

- Direction (i) is motivated by the fact that tuning hyperparameters is usually impossible for unsupervised scenarios in which labeled data are rare. Nonetheless, performance of the proposed the VFA-based framework is highly sensitive to the chosen wavelet mother kernel and wavelet level. Therefore, it would be interesting to let the ML algorithm learn the wavelet level implicitly.

- The idea behind direction (ii) is that common anomaly detection approaches relies on training a predictive model using only normal data, thus new data points are classified as anomalies if their actual values deviate from the values predicted by the model more than a given threshold. Therefore VFA coefficients can be fed to the predictive model and nodes with VFA coefficients differing from the output would be assigned as anomalies. Also, the proposed SG-GRU can be used as the predictive model in the anomaly detector.

- Theorem 5.2.4 shows that $K = \mathcal{F}$, the size of the GS spectral support, should be at most $M = |\mathcal{S}|$, the size of its vertex support. Nonetheless, using values of K close to M accumulates errors throughout the learning model regarding the actual bandwidth of the GS. Therefore, it would be interesting to substantiate the selection of $\mathcal{F}$ and provide error bounds for the model based on prior assumptions of the GS.

Referências Bibliográficas

[1] AKELLA, P. N., CONNORS, T. J., KELLY, J., et al. "Creation and maintenance of social relationship network graphs", *Google Patents*, May 2009.

[2] CROVELLA, M., KOLACZYK, E. "Graph wavelets for spatial traffic analysis". In: *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, v. 3, pp. 1848–1857 vol.3, Mar 2003.

[3] HAENGGI, M., ANDREWS, J. G., BACCELLI, F., et al. "Stochastic geometry and random graphs for the analysis and design of wireless networks", *IEEE Journal on Selected Areas in Communications*, v. 27, n. 7, pp. 1029–1046, Aug 2009.

[4] CHAOUIYA, C., REMY, E., RUET, P., et al. "Qualitative modelling of genetic networks: From logical regulatory graphs to standard Petri nets". In: *International Conference on Application and Theory of Petri Nets*, pp. 137–156. Springer, 2004.

[5] GOLDSBERRY, L., HUANG, W., WYMBS, N. F., et al. "Brain signal analytics from graph signal processing perspective", *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 851–855, 2017.

[6] SHUMAN, D. I., NARANG, S. K., FROSSARD, P., et al. "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains", *IEEE Signal Processing Magazine*, v. 30, n. 3, pp. 83–98, May 2013.

[7] HUSSAIN, S. F., HARIS, M. "A k-means based co-clustering (kCC) algorithm for sparse, high dimensional data", *Expert Systems with Applications*, v. 118, pp. 20–34, Mar 2019.

[8] WAGNER, R., CHOI, H., BARANIUK, R., et al. "Distributed wavelet transform for irregular sensor network grids". In: *IEEE/SP 13th Workshop on Statistical Signal Processing, 2005*, pp. 1196–1201. IEEE, 2005.

[9] WAGNER, R. S., BARANIUK, R. G., DU, S., et al. "An architecture for distributed wavelet analysis and processing in sensor networks". In: *Proceedings of the 5th international conference on Information processing in sensor networks*, pp. 243–250. ACM, 2006.

[10] CIANCIO, A., PATTEM, S., ORTEGA, A., et al. "Energy-efficient data representation and routing for wireless sensor networks based on a distributed wavelet compression algorithm". In: *Proceedings of the 5th international conference on Information processing in sensor networks*, pp. 309–316. ACM, 2006.

[11] SHEN, G., ORTEGA, A. "Joint routing and 2D transform optimization for irregular sensor network grids using wavelet lifting". In: *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*, pp. 183–194. IEEE, 2008.

[12] ZHU, X., RABBAT, M. "Graph spectral compressed sensing for sensor networks". In: *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2865–2868. IEEE, 2012.

[13] KANEKO, M., CHEUNG, G., SU, W.-T., et al. "Graph-based joint signal/power restoration for energy harvesting wireless sensor networks". In: *GLOBECOM 2017-2017 IEEE Global Communications Conference*, pp. 1–6. IEEE, 2017.

[14] EGILMEZ, H. E., ORTEGA, A. "Spectral anomaly detection using graph-based filtering for wireless sensor networks". In: *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 1085–1089. IEEE, 2014.

[15] JAIN, R. K., MOURA, J. M., KONTOKOSTA, C. E. "Big data+ big cities: Graph signals of urban air pollution", *IEEE Signal Processing Magazine*, v. 31, n. 5, pp. 130–136, 2014.

[16] JABŁOŃSKI, I. "Graph signal processing in applications to sensor networks, smart grids, and smart cities", *IEEE Sensors Journal*, v. 17, n. 23, pp. 7659–7666, 2017.

[17] DONG, X., ORTEGA, A., FROSSARD, P., et al. "Inference of mobility patterns via spectral graph wavelets". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 3118–3122. IEEE, 2013.

[18] ATASOY, S., DONNELLY, I., PEARSON, J. “Human brain networks function in connectome-specific harmonic waves”, *Nature communications*, v. 7, pp. 10340, 2016.

[19] GRIFFA, A., RICAUD, B., BENZI, K., et al. “Transient networks of spatio-temporal connectivity map communication pathways in brain functional systems”, *NeuroImage*, v. 155, pp. 490–502, 2017.

[20] RUI, L., NEJATI, H., CHEUNG, N.-M. “Dimensionality reduction of brain imaging data using graph signal processing”. In: *2016 IEEE International Conference on Image Processing (ICIP)*, pp. 1329–1333. IEEE, 2016.

[21] JESTROVIĆ, I., COYLE, J. L., SEJDIĆ, E. “Differences in brain networks during consecutive swallows detected using an optimized vertex–frequency algorithm”, *Neuroscience*, v. 344, pp. 113–123, 2017.

[22] GUO, Y., NEJATI, H., CHEUNG, N.-M. “Deep neural networks on graph signals for brain imaging analysis”. In: *2017 IEEE International Conference on Image Processing (ICIP)*, pp. 3295–3299. IEEE, 2017.

[23] HUANG, W., BOLTON, T. A., MEDAGLIA, J. D., et al. “A graph signal processing perspective on functional brain imaging”, *Proceedings of the IEEE*, v. 106, n. 5, pp. 868–885, 2018.

[24] ANIRUDH, R., THIAGARAJAN, J. J. “Bootstrapping graph convolutional neural networks for autism spectrum disorder classification”. In: *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3197–3201. IEEE, 2019.

[25] HU, C., SEPULCRE, J., JOHNSON, K. A., et al. “Matched signal detection on graphs: Theory and application to brain imaging data classification”, *NeuroImage*, v. 125, pp. 587–600, 2016.

[26] HU, C., HUA, X., YING, J., et al. “Localizing sources of brain disease progression with network diffusion model”, *IEEE journal of selected topics in signal processing*, v. 10, n. 7, pp. 1214–1225, 2016.

[27] NARANG, S. K., GADDE, A., ORTEGA, A. “Signal processing techniques for interpolation in graph structured data”. In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 5445–5449, May 2013.

[28] RIBEIRO, G., LIMA, J. “Graph Signal Processing in a Nutshell”, *Journal of Communication and Information Systems*, v. 33, n. 1, Jul 2018.

[29] NARANG, S. K., ORTEGA, A. "Downsampling graphs using spectral theory". In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 4208–4211, Mar 2011.

[30] HAMMOND, D. K., VANDERGHEYNST, P., GRIBONVAL, R. "Wavelets on graphs via spectral graph theory", *Applied and Computational Harmonic Analysis*, v. 30, n. 2, pp. 129–150, 2011.

[31] PÜSCHEL, M., MOURA, J. M. "Algebraic signal processing theory", *preprint cs/0612077*, 2006.

[32] AGGARWAL, C. C. *Outlier analysis*. Springer, 2013.

[33] SANDRYHAILA, A., MOURA, J. M. "Discrete signal processing on graphs: Frequency analysis", *IEEE Transactions on Signal Processing*, v. 62, n. 12, pp. 3042–3054, 2014.

[34] MASOUMI, M., HAMZA, A. B. "Spectral shape classification: A deep learning approach", *Journal of Visual Communication and Image Representation*, v. 43, pp. 198–211, Feb 2017.

[35] VALDIVIA, P., DIAS, F., PETRONETTO, F., et al. "Wavelet-based visualization of time-varying data on graphs". In: *2015 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pp. 1–8, Oct 2015.

[36] DAL COL, A., VALDIVIA, P., PETRONETTO, F., et al. "Wavelet-based visual analysis of dynamic networks", *IEEE transactions on visualization and computer graphics*, v. 24, n. 8, pp. 2456–2469, 2017.

[37] SALMAN, A. G., KANIGORO, B., HERYADI, Y. "Weather forecasting using deep learning techniques". In: *International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 281–285, Oct 2015.

[38] LV, Y., DUAN, Y., KANG, W., et al. "Traffic Flow Prediction With Big Data: A Deep Learning Approach", *IEEE Transactions on Intelligence Transportation Systems*, v. 16, n. 2, pp. 865–873, Apr 2015.

[39] SMITH, S. M., MILLER, K. L., SALIMI-KHORSHIDI, G., et al. "Network modelling methods for FMRI", *NeuroImage*, v. 54, n. 2, pp. 875–891, Jan 2011. ISSN: 1053-8119.

[40] LI, L., OTA, K., DONG, M. "When Weather Matters: IoT-Based Eletric. Load Forecasting for Smart Grid", *IEEE Communications Magazine*, v. 55, n. 10, pp. 46–51, Oct 2017.

[41] RACAH, E., BECKHAM, C., MAHARAJ, T., et al. "ExtremeWeather: A large-scale climate dataset for semi-supervised detection, localization, and understanding of extreme weather events". In: Guyon, I., Luxburg, U. V., Bengio, S., et al. (Eds.), *Advances in Neural Information Processing Systems 30 (NIPS)*, Curran Associates, Inc., pp. 3402–3413, Dec 2017.

[42] WILLIAMS, B. M., HOEL, L. A. "Modeling and forecasting vehicular traffic flow as a seasonal ARIMA process: Theoretical basis and empirical results", *Journal of Transportation Engineering*, v. 129, n. 6, pp. 664–672, Oct 2003.

[43] CAI, L., ZHANG, Z., YANG, J., et al. "A noise-immune Kalman filter for short-term traffic flow forecasting", *Physica A: Statistical Mechanics and its Applications*, v. 536, pp. 122601, Dec 2019. ISSN: 0378-4371.

[44] CHANDRA, S. R., AL-DEEK, H. "Predictions of freeway traffic speeds and volumes using vector autoregressive models", *Journal of Intelligence Transportation Systems*, v. 13, n. 2, pp. 53–72, Apr 2009.

[45] KRAMER, O., GIESEKE, F. "Short-term wind energy forecasting using support vector regression". In: *Soft Computer Models in Industrial and Environmental Applications,* 6th *International Conference (SOCO)*, v. 87, pp. 271–280. Springer, 2011.

[46] LESHEM, G., RITOV, Y. "Traffic flow prediction using AdaBoost algorithm with random forests as a weak learner", *International Journal of Intelligence Technology*, v. 2, pp. 1305–6417, Jan 2007.

[47] HUANG, W., SONG, G., HONG, H., et al. "Deep architecture for traffic flow prediction: deep belief networks with multitask learning", *IEEE Transactions on Intelligence Transportation Systems*, v. 15, n. 5, pp. 2191–2201, Apr 2014.

[48] YUHAN JIA, JIANPING WU, YIMAN DU. "Traffic speed prediction using deep learning method". In: *IEEE International Conference on Intelligence Transportation Systems (ITSC)*, pp. 1217–1222, Dec 2016.

[49] SALMAN, A., HERYADI, Y., ABDURAHMAN, E., et al. "Weather forecasting using merged Long Short-Term Memory Model (LSTM) and Autoregressive Integrated Moving Average (ARIMA) Model", *Journal of Comput. Science*, v. 14, pp. 930–938, July 2018.

[50] LIANG, S., NGUYEN, L., JIN, F. “A Multi-variable Stacked Long-Short Term Memory Network for Wind Speed Forecasting”. In: *IEEE International Conference on Big Data*, pp. 4561–4564, Dec 2018.

[51] GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep learning.* United States of America, MIT press, 2016.

[52] SHI, X., CHEN, Z., WANG, H., et al. “Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting”. In: Cortes, C., Lawrence, N. D., Lee, D. D., et al. (Eds.), *Advances in Neural Information Processing Systems 28*, Curran Associates, Inc., pp. 802–810, 2015.

[53] YU, H., WU, Z., WANG, S., et al. “Spatiotemporal Recurrent Convolutional Networks for Traffic Prediction in Transportation Networks”, *Sensors*, v. 17, n. 7, pp. 1–16, June 2017. ISSN: 14248220.

[54] HUANG, C. J., KUO, P. H. “A deep CNN-LSTM model for particulate matter ($\mathrm{Pm}_{2.5}$) forecasting in smart cities”, *Sensors*, v. 18, n. 2220, July 2018.

[55] WU, Y., TAN, H., QIN, L., et al. “A hybrid deep learning based traffic flow prediction method and its understanding”, *Transportation Research Part C: Emerging Technologies*, v. 90, pp. 166–180, May 2018.

[56] LI, W., TAO, W., QIU, J., et al. “Densely Connected Convolutional Networks With Attention LSTM for Crowd Flows Prediction”, *IEEE Access*, v. 7, pp. 140488–140498, Sep 2019.

[57] HENAFF, M., BRUNA, J., LECUN, Y. “Deep convolutional networks on graph-structured data”, *eprint arXiv:1506.05163*, June 2015.

[58] DEFFERRARD, M., BRESSON, X., VANDERGHEYNST, P. “Convolutional neural networks on graphs with fast localized spectral filtering”. In: *Advances in neural information processing systems*, pp. 3844–3852, 2016.

[59] LEVIE, R., MONTI, F., BRESSON, X., et al. “CayleyNets: Graph Convolutional Neural Networks With Complex Rational Spectral Filters”, *IEEE Transactions on Signal Processing*, v. 67, n. 1, pp. 97–109, Nov 2019.

[60] KIPF, T. N., WELLING, M. “Semi-supervised classification with graph convolutional networks”, *eprint arXiv:1609.02907*, Sep 2016.

[61] ORTEGA, A., FROSSARD, P., KOVAČEVIĆ, J., et al. “Graph Signal Processing: Overview, Challenges, and Applications”, *Proceedings of the IEEE*, v. 106, n. 5, pp. 808–828, May 2018.

[62] NARANG, S. K., GADDE, A., SANOU, E., et al. “Localized iterative methods for interpolation in graph structured data”. In: *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 491–494, Dec 2013.

[63] CHEN, J., FANG, H.-R., SAAD, Y. “Fast Approximate kNN Graph Construction for High Dimensional Data via Recursive Lanczos Bisection.” *Journal of Machine Learning Research*, v. 10, n. 9, Sep 2009.

[64] SEGARRA, S., MARQUES, A. G., LEUS, G., et al. “Interpolation of graph signals using shift-invariant graph filters”. In: *23rd European Signal Processing Conference (EUSIPCO)*, pp. 210–214, Aug 2015.

[65] GAVISH, M., NADLER, B., COIFMAN, R. R. “Multiscale Wavelets on Trees, Graphs and High Dimensional Data: Theory and Applications to Semi Supervised Learning.” In: *International Conference on Machine Learning (ICML)*, pp. 367–374, 2010.

[66] ANIS, A., GADDE, A., ORTEGA, A. “Towards a sampling theorem for signals on arbitrary graphs”. In: *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3864–3868, May 2014.

[67] CHEN, S., CERDA, F., RIZZO, P., et al. “Semi-Supervised Multiresolution Classification Using Adaptive Graph Filtering With Application to Indirect Bridge Structural Health Monitoring”, *IEEE Transactions on Signal Processing*, v. 62, n. 11, pp. 2879–2893, Mar 2014.

[68] CHEN, S., VARMA, R., SANDRYHAILA, A., et al. “Discrete Signal Processing on Graphs: Sampling Theory”, *IEEE Transactions on Signal Processing*, v. 63, n. 24, pp. 6510–6523, Aug 2015.

[69] NARANG, S. K., ORTEGA, A. “Local two-channel critically sampled filter-banks on graphs”, *IEEE International Conference on Image Processing*, pp. 333–336, Sep 2010.

[70] SPELTA, M. J., MARTINS, W. A. “Online temperature estimation using graph signals”, *XXXVI Simpósio Brasileiro de Telecomunicaçoes e Processamento de Sinais (SBrT)*, pp. 154–158, Sep 2018.

[71] SPELTA, M. J. M., MARTINS, W. A. “Normalized LMS algorithm and data-selective strategies for adaptive graph signal estimation”, *Signal Processing*, v. 167, pp. 107326, Feb 2020.

[72] ANIS, A., GADDE, A., ORTEGA, A. “Efficient Sampling Set Selection for Bandlimited Graph Signals Using Graph Spectral Proxies”, *IEEE Transactions on Signal Processing*, v. 64, n. 14, pp. 3775–3789, Mar 2016.

[73] CHAMON, L. F. O., RIBEIRO, A. “Greedy Sampling of Graph Signals”, *IEEE Transactions on Signal Processing*, v. 66, n. 1, pp. 34–47, Jan 2018.

[74] SHUMAN, D. I., FARAJI, M., VANDERGHEYNST, P. “Semi-Supervised Learning with Spectral Graph Wavelets”. In: *Proceedings of the International Conference on Sampling Theory and Applications (SampTA)*, May 2011.

[75] GADDE, A., ANIS, A., ORTEGA, A. “Active Semi-Supervised Learning Using Sampling Theory for Graph Signals”. In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD ’14, p. 492–501. Association for Computing Machinery, 2014.

[76] CHEN, S., VARMA, R., SINGH, A., et al. “Signal Recovery on Graphs: Fundamental Limits of Sampling Strategies”, *IEEE Transactions on Signal and Information Processing over Networks*, v. 2, n. 4, pp. 539–554, Oct 2016.

[77] PARISOT, S., KTENA, S. I., FERRANTE, E., et al. “Disease prediction using graph convolutional networks: Application to Autism Spectrum Disorder and Alzheimer’s disease”, *Medical Image Analysis*, v. 48, pp. 117 – 130, Aug 2018.

[78] CHEUNG, G., SU, W., MAO, Y., et al. “Robust Semi-supervised Graph Classifier Learning With Negative Edge Weights”, *IEEE Transactions on Signal and Information Processing over Networks*, v. 4, n. 4, pp. 712–726, Mar 2018.

[79] HUANG, W., MARQUES, A. G., RIBEIRO, A. “Collaborative filtering via graph signal processing”. In: 25th *European Signal Processing Conference (EUSIPCO)*, pp. 1094–1098, Sep 2017.

[80] HUANG, W., MARQUES, A. G., RIBEIRO, A. R. “Rating Prediction via Graph Signal Processing”, *IEEE Transactions on Signal Processing*, v. 66, n. 19, pp. 5066–5081, Aug 2018.

[81] MANOHAR, K., BRUNTON, B. W., KUTZ, J. N., et al. “Data-Driven Sparse Sensor Placement for Reconstruction: Demonstrating the Benefits of Exploiting Known Patterns”, *IEEE Control Systems Magazine*, v. 38, n. 3, pp. 63–86, June 2018.

[82] XU, B., SHEN, H., CAO, Q., et al. "Graph wavelet neural network", 7^{th} *International Conference on Learning Representations (ICLR)*, pp. 1–13, Apr 2019.

[83] ROMERO, D., IOANNIDIS, V. N., GIANNAKIS, G. B. "Kernel-Based Reconstruction of Space-Time Functions on Dynamic Graphs", *IEEE Journal on Selected Topics in Signal Processing*, v. 11, n. 6, pp. 856–869, Sep 2017.

[84] KHODAYAR, M., WANG, J. "Spatio-Temporal Graph Deep Neural Network for Short-Term Wind Speed Forecasting", *IEEE Transactions on Sustainable Energy*, v. 10, n. 2, pp. 670–681, Apr 2019.

[85] STANKOVIĆ, L., SEJDIĆ, E. *Vertex-Frequency Analysis of Graph Signals.* Springer, 2019.

[86] PESENSON, I. "Sampling in Paley-Wiener spaces on combinatorial graphs", *Transactions of the American Mathematical Society*, v. 360, n. 10, pp. 5603–5627, May 2008.

[87] WU, Z., PAN, S., CHEN, F., et al. "A comprehensive survey on graph neural networks", *IEEE Transactions on Neural Networks and Learning Systems*, Mar 2020.

[88] TORBJØRN T. "Draw a graph in Latex with Tikz". Disponível em: <`"https://tex.stackexchange.com/questions/270543/draw-a-graph-in-latex-with-tikz"`>. Acesso em: (2020, Feb. 1).

[89] CHEBOTAREV, P. "The walk distances in graphs", *Discrete Applied Mathematics*, v. 160, n. 10-11, pp. 1484–1500, 2012.

[90] GALLIER, J. "Notes on elementary spectral graph theory. applications to graph clustering using normalized cuts", *preprint arXiv:1311.2492*, Nov 2013.

[91] WARDETZKY, M., MATHUR, S., KÄLBERER, F., et al. "Discrete Laplace operators: no free lunch". In: *Symposium on Geometry processing*, pp. 33–37. Aire-la-Ville, Switzerland, 2007.

[92] AXLER, S. *Linear algebra done right.* Cham, Springer, 2015.

[93] SANDRYHAILA, A., MOURA, J. M. F. "Discrete Signal Processing on Graphs", *IEEE Transactions on Signal Processing*, v. 61, n. 7, pp. 1644–1656, Jan 2013.

[94] HARTFIEL, D. J. "Dense sets of diagonalizable matrices", *Proceedings of the American Mathematical Society*, v. 123, n. 6, pp. 1669–1672, 1995.

[95] SHAFIPOUR, R., KHODABAKHSH, A., MATEOS, G., et al. "A digraph Fourier transform with spread frequency components". In: *2017 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 583–587. IEEE, 2017.

[96] SANDRYHAILA, A., MOURA, J. M. F. "Discrete signal processing on graphs: Graph filters". In: *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6163–6166, 2013.

[97] LI, S., JIN, Y., SHUMAN, D. I. "Scalable M-Channel Critically Sampled Filter Banks for Graph Signals", *preprint arXiv:1608.03171*, 2016.

[98] ELIAS, V. R. M., MARTINS, W. A. "Graph Fourier transform for light field compression", *XXXV Simpósio Brasileiro de Telecomunições e Processamento de Sinais (SBrT), São Pedro, São Paulo*, 2017.

[99] ELIAS, V. R. M., MARTINS, W. A. "On the use of graph Fourier transform for light-field compression", *Journal of Communication and Information Systems*, v. 33, n. 1, May 2018.

[100] BERGER, P., HANNAK, G., MATZ, G. "Efficient graph learning from noisy and incomplete data", *IEEE Transactions on Signal and Information Processing over Networks*, v. 6, pp. 105–119, Jan 2020.

[101] SHUMAN, D. I., RICAUD, B., VANDERGHEYNST, P. "A windowed graph Fourier transform". In: *2012 IEEE Statistical Signal Processing Workshop (SSP)*, pp. 133–136. Ieee, 2012.

[102] RABIEI, H., RICHARD, F., COULON, O., et al. "Estimating the Complexity of the Cerebral Cortex Folding with a Local Shape Spectral Analysis". In: *Vertex-Frequency Analysis of Graph Signals*, Springer, Cham, pp. 437–458, Dec 2019.

[103] DEFFERRARD, M., MARTIN, L., PENA, R., et al. "PYGSP: Graph Signal Processing in Python. [Online]". Disponível em: <`https://github.com/epfl-lts2/pygsp/`>. Acesso em: (2020, Feb. 1).

[104] MALLAT, S. *A wavelet tour of signal processing.* Elsevier, 1999.

[105] COIFMAN, R. R., MAGGIONI, M. "Diffusion wavelets", *Applied and Computational Harmonic Analysis*, v. 21, n. 1, pp. 53–94, 2006.

[106] MAGGIONI, M., BREMER JR, J. C., COIFMAN, R. R., et al. "Biorthogonal diffusion wavelets for multiscale representation on manifolds and graphs". In: *Wavelets XI*, v. 5914, p. 59141M. International Society for Optics and Photonics, 2005.

[107] NARANG, S. K., ORTEGA, A. "Lifting based wavelet transforms on graphs". In: *Proceedings: APSIPA ASC 2009: Asia-Pacific Signal and Information Processing Association, 2009 Annual Summit and Conference*, pp. 441–444, Oct 2009.

[108] DINIZ, P. S., DA SILVA, E. A., NETTO, S. L. *Digital signal processing: system analysis and design.* Cambridge, Cambridge University Press, 2010.

[109] LEONARDI, N., VAN DE VILLE, D. "Tight wavelet frames on multislice graphs", *IEEE Transactions on Signal Processing*, v. 61, n. 13, pp. 3357–3367, 2013.

[110] SHUMAN, D. I., WIESMEYR, C., HOLIGHAUS, N., et al. "Spectrum-adapted tight graph wavelet and vertex-frequency frames", *IEEE Transactions on Signal Processing*, v. 63, n. 16, pp. 4223–4235, 2015.

[111] JIN, Y., SHUMAN, D. I. "An M-channel critically sampled filter bank for graph signals", pp. 3909–3913, Mar 2017.

[112] SAKIYAMA, A., WATANABE, K., TANAKA, Y. "Spectral graph wavelets and filter banks with low approximation error", *IEEE Transactions on Signal and Information Processing over Networks*, v. 2, n. 3, pp. 230–245, 2016.

[113] SHUMAN, D. I., VANDERGHEYNST, P., FROSSARD, P. "Chebyshev polynomial approximation for distributed signal processing". In: *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pp. 1–8. IEEE, 2011.

[114] CHAPELLE, O., SCHÖLKOPF, B., ZIEN, A. *Semi-supervised learning.* Cambridge, Massachusetts, The MIT Press, 2006.

[115] SHANNON, C. E. "A mathematical theory of communication", *Bell system technical journal*, v. 27, n. 3, pp. 379–423, 1948.

[116] LORENZO, P., BARBAROSSA, S., BANELLI, P. "Sampling and Recovery of Graph Signals". In: *Cooperative and Graph Signal Processing*, Academic Press, pp. 261 – 282, Italy, 2018.

[117] PUY, G., TREMBLAY, N., GRIBONVAL, R., et al. "Random sampling of bandlimited signals on graphs", *Applied and Computational Harmonic Analysis*, v. 44, n. 2, pp. 446–475, 2018.

[118] DENG, L. "A tutorial survey of architectures, algorithms, and applications for deep learning", *APSIPA Transactions on Signal and Information Processing*, v. 3, Jan 2014.

[119] YANG, Y., YE, Z., SU, Y., et al. "Deep learning for in vitro prediction of pharmaceutical formulations", *Acta pharmaceutica sinica B*, v. 9, n. 1, pp. 177–185, 2019.

[120] NIV, S., GORDON, G., NATAN, A. "Deep Learning embedding layers for better prediction of atomic forces in solids", *Bulletin of the American Physical Society*, v. 65, n. 1, Mar 2020.

[121] GLASER, J. I., BENJAMIN, A. S., FARHOODI, R., et al. "The roles of supervised machine learning in systems neuroscience", *Progress in neurobiology*, v. 175, pp. 126–137, Apr 2019.

[122] EMERSON, S., KENNEDY, R., O'SHEA, L., et al. "Trends and applications of machine learning in quantitative finance". In: *8th International Conference on Economics and Finance Research (ICEFR 2019)*, 2019.

[123] HOCHREITER, S., SCHMIDHUBER, J. "Long short-term memory", *Neural computation MIT press*, v. 9, n. 8, pp. 1735–1780, Nov 1997.

[124] CHUNG, J., GULCEHRE, C., CHO, K., et al. "Empirical evaluation of gated recurrent neural networks on sequence modeling", *arXiv:1412.3555*, Dec 2014.

[125] BAI, S., KOLTER, J. Z., KOLTUN, V. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling", *preprint arXiv:1803.01271*, Mar 2018.

[126] V., J. L. Disponível em: <`https://tex.stackexchange.com/questions/432312/how-do-i-draw-an-lstm-cell-in-tikz/432344`>.

[127] BLOG, W. "LSTMs: The Gentle Giants". 2017. Disponível em: <`https://weberna.github.io/blog/2017/11/15/LSTM-Vanishing-Gradients.html#fnref:3`>.

[128] GRAVES, A. "Generating sequences with recurrent neural networks", *eprint arXiv:1308.0850*, Aug 2013.

[129] HINTON, G., SRIVASTAVA, N., SWER-SKY, K. “lecture6a overview of mini-batch gradient descent”. 2012. Disponível em: <https://www.cs.toronto.edu/~tijmen/csc321/slides/lecture_slides_lec6.pdf>.

[130] BRUNA, J., ZAREMBA, W., SZLAM, A., et al. “Spectral networks and locally connected networks on graphs”, *eprint arXiv:1312.6203*, Dec 2013.

[131] LI, Y., YU, R., SHAHABI, C., et al. “Diffusion convolutional recurrent neural network: Data-driven traffic forecasting”. In: *6*th *International Conference on Learning Representations (ICLR)*, pp. 1–16, Fev 2018.

[132] VON LUXBURG, U. “A tutorial on spectral clustering”, *Statistics and computing*, v. 17, n. 4, pp. 395–416, Aug 2007.

[133] DHILLON, I. S., GUAN, Y., KULIS, B. “Weighted Graph Cuts without Eigenvectors A Multilevel Approach”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, v. 29, n. 11, pp. 1944–1957, Sep 2007.

[134] FEY, M. “PyTorchGeometric”. Disponível em: <https://pytorch-geometric.readthedocs.io/en/latest/>.

[135] WANG, M., YU, L., GAN, Q., et al. “Deep Graph Library”. Disponível em: <https://docs.dgl.ai/en/0.4.x/index.html>.

[136] MORRIS, C., KRIEGE, N. M., BAUSE, F., et al. “TUDataset: A collection of benchmark datasets for learning with graphs”, *eprint: ArXiv:2007.08663*, Jul 2020.

[137] MORRIS, C., KRIEGE, N. M., BAUSE, F., et al. “TUDatasets”. Disponível em: <https://chrsmrrs.github.io/datasets/>.

[138] CHANDOLA, V., BANERJEE, A., KUMAR, V. “Anomaly detection: A survey”, *ACM computing surveys (CSUR)*, v. 41, n. 3, pp. 15, July 2009.

[139] HAWKINS, D. M. *Identification of outliers*, v. 11. London, Springer, 1980.

[140] “Premature atrial contraction”. Disponível em: <https://ecgwaves.com/topic/premature-atrial-contraction-beat-complex/>. Acesso em: (2020, July 20).

[141] LEWENFUS, G., ALVES MARTINS, W., CHATZINOTAS, S., et al. “On the Use of Vertex-Frequency Analysis for Anomaly Detection in Graph Signals”, *Anais do XXXVII Simpósio Brasileiro de Telecomunicações e Processamento de Sinais (SBrT 2019)*, pp. 1–5, 2019.

[142] BREUNIG, M. M., KRIEGEL, H.-P., NG, R. T., et al. "LOF: identifying density-based local outliers". In: *ACM sigmod record*, v. 29, pp. 93–104. ACM, May 2000.

[143] LIU, F. T., TING, K. M., ZHOU, Z.-H. "Isolation forest". In: *2008 Eighth IEEE International Conference on Data Mining*, pp. 413–422. IEEE, 2008.

[144] SCHÖLKOPF, B., WILLIAMSON, R. C., SMOLA, A. J., et al. "Support vector method for novelty detection", *Advances in neural information processing systems*, pp. 582–588, 2000.

[145] INSTITUTO NACIONAL DE METEOROLOGIA (IMET). "Normais Climatológicas do Brasil". Disponível em: <`http://www.inmet.gov.br/portal/index.php?r=clima/normaisClimatologicas`>. Acesso em: (2018, Feb. 1).

[146] RASMUSSEN, C. E., WILLIAMS, C. K. I. *Gaussian Processes for Machine Learning*. Berlin, Heidelberg, the MIT Press, 2006.

[147] BUHMANN, M. D. "Radial basis functions", *Acta numerica*, v. 9, pp. 1–38, Jan 2000.

[148] ZHAO, L., SONG, Y., DENG, M., et al. "Temporal graph convolutional network for urban traffic flow prediction method", *eprint arXiv:1811.05320*, Dec 2018.

[149] ZHANG, J., SHI, X., XIE, J., et al. "GaAN: gated attention networks for learning on large and spatiotemporal graphs", *eprint arXiv:1803.07294*, Mar 2018.

[150] WANG, B., LUO, X., ZHANG, F., et al. "Graph-based deep modeling and real time forecasting of sparse spatio-temporal data", *eprint arXiv:1804.00684*, Apr 2018.

[151] GENG, X., LI, Y., WANG, L., et al. "Spatiotemporal Multi-Graph Convolution Network for Ride-Hailing Demand Forecasting", *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 33, n. 01, pp. 3656–3663, July 2019.

[152] CHEN, W., CHEN, L., XIE, Y., et al. "Multi-Range Attentive Bicomponent Graph Convolutional Network for Traffic Forecasting", *eprint arXiv:1911.12093*, Nov 2019.

[153] CUI, Z., HENRICKSON, K., KE, R., et al. "Traffic Graph Convolutional Recurrent Neural Network: A Deep Learning Framework for Network-Scale Traffic Learning and Forecasting", *IEEE Transactions on Intelligence Transportation Systems*, Nov 2019.

[154] YU, B., YIN, H., ZHU, Z. "Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting". In: *Proceedings of the 27*th *International Joint Conference on Artificial Intelligence*, pp. 3634–3640, July 2018.

[155] WANG, M., LAI, B., JIN, Z., et al. "Dynamic spatio-temporal graph-based cnns for traffic prediction", *eprint arXiv:1812.02019*, Mar 2020.

[156] WU, Z., PAN, S., LONG, G., et al. "Graph WaveNet for deep spatial-temporal graph modeling", *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 1907–1913, Aug 2019.

[157] LEE, K., RHEE, W. "DDP-GCN: Multi-Graph Convolutional Network for Spatiotemporal Traffic Forecasting", *eprint arXiv:1905.12256*, May 2019.

[158] DIAO, Z., WANG, X., ZHANG, D., et al. "Dynamic spatial-temporal graph convolutional neural networks for traffic forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 33, pp. 890–897, July 2019.

[159] FANG, S., ZHANG, Q., MENG, G., et al. "GstNet: Global spatial-temporal network for traffic flow prediction". In: *Proceedings of the 28*th *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 10–16, Aug 2019.

[160] SONG, C., LIN, Y., GUO, S., et al. "Spatial-Temporal Synchronous Graph Convolutional Networks: A New Framework for Spatial-Temporal Network Data Forecasting". 2020. Disponível em: <`https://github.com/wanhuaiyu/STSGCN/blob/master/paper/AAAI2020-STSGCN.pdf`>.

[161] GUO, S., LIN, Y., FENG, N., et al. "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting". In: *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 33, pp. 922–929, July 2019.

[162] CUI, Z., KE, R., PU, Z., et al. "Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction", *eprint arXiv:1801.02143*, Jan 2018.

[163] WANG, X., CHEN, C., MIN, Y., et al. "Efficient metropolitan traffic prediction based on graph recurrent neural network", *arXiv:1811.00740*, Nov 2018.

[164] LIAO, B., MCILWRAITH, D., ZHANG, J., et al. "Deep sequence learning with auxiliary information for traffic prediction", *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 537–546, July 2018.

[165] LIAO, B., ZHANG, J., CAI, M., et al. "Dest-ResNet: A Deep Spatiotemporal Residual Network for Hotspot Traffic Speed Prediction". In: *Proceedings of the* 26th *ACM International Conference on Multimedia*, pp. 1883–1891, Oct 2018.

[166] HAN, D., CHEN, J., SUN, J. "A parallel spatiotemporal deep learning network for highway traffic flow forecasting", *International Journal of Distributed Sensor Networks*, v. 15, n. 2, Feb 2019.

[167] LV, Z., XU, J., ZHENG, K., et al. "LC-RNN: A deep learning model for traffic speed prediction". In: *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3470–3476, July 2018.

[168] LIN, Y., DAI, X., LI, L., et al. "Pattern Sensitive Prediction of Traffic Flow Based on Generative Adversarial Framework", *IEEE Transactions on Intelligence Transportation Systems*, v. 20, n. 6, pp. 2395–2400, Aug 2019.

[169] YANG, H. F., DILLON, T. S., CHEN, Y. P. P. "Optimized Structure of the Traffic Flow Forecasting Model with a Deep Learning Approach", *IEEE Transactions on Neural Networks and Learning Systems*, v. 28, n. 10, pp. 2371–2381, Oct 2017.

[170] ZHANG, K., ZHENG, L., LIU, Z., et al. "A deep learning based multitask model for network-wide traffic speed prediction", *Neurocomputing*, v. 396, pp. 438 – 450, Apr 2020. ISSN: 0925-2312.

[171] OUYANG, X., ZHANG, C., ZHOU, P., et al. "Deepspace: An online deep learning framework for mobile big data to understand human mobility patterns", *arXiv:1610.07009*, Mar 2016.

[172] KHODAYAR, M., KAYNAK, O., KHODAYAR, M. E. "Rough Deep Neural Architecture for Short-Term Wind Speed Forecasting", *IEEE Transactions on Ind. Inform.*, v. 13, n. 6, pp. 2770–2779, July 2017.

[173] ZHU, Q., CHEN, J., ZHU, L., et al. "Wind speed prediction with spatio-temporal correlation: A deep learning approach", *Energies*, v. 11, pp. 705, Mar 2018.

[174] ZHANG, C. Y., CHEN, C. L., GAN, M., et al. "Predictive Deep Boltzmann Machine for Multiperiod Wind Speed Forecasting", *IEEE Transactions on Sustainable Energy*, v. 6, n. 4, pp. 1416–1425, July 2015. ISSN: 19493029. doi: 10.1109/TSTE.2015.2434387.

[175] KHODAYAR, M., MOHAMMADI, S., KHODAYAR, M. E., et al. "Convolutional Graph Autoencoder: A Generative Deep Neural Network for Probabilistic Spatio-Temporal Solar Irradiance Forecasting", *IEEE Transactions on Sustainable Energy*, v. 11, n. 2, pp. 571–583, Apr 2020.

[176] RASP, S., LERCH, S. "Neural networks for postprocessing ensemble weather forecasts", *Monthly Weather Review*, v. 146, n. 11, pp. 3885–3900, Oct 2018.

[177] LI, C., CUI, Z., ZHENG, W., et al. "Spatio-temporal graph convolution for skeleton based action recognition". In: *Proceedings of the AAAI Conference on Artificial Intelligence*, Apr 2018.

[178] WU, C., WU, X.-J., KITTLER, J. "Spatial Residual Layer and Dense Connection Block Enhanced Spatial Temporal Graph Convolutional Network for Skeleton-Based Action Recognition". In: *The IEEE International Conference on Comput. Vision (ICCV) Workshops*, Oct 2019.

[179] SI, C., CHEN, W., WANG, W., et al. "An Attention Enhanced Graph Convolutional LSTM Network for Skeleton-Based Action Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[180] GADGIL, S., ZHAO, Q., ADELI, E., et al. "Spatio-Temporal Graph Convolution for Functional MRI Analysis", *eprint arXiv:2003.10613*, Mar 2020.

[181] HUANG, H., HU, X., HAN, J., et al. "Latent source mining in FMRI data via deep neural network", *IEEE International Symposium on Biomedical Imaging (ISBI)*, pp. 638–641, July 2016. ISSN: 19458452. doi: 10.1109/ISBI.2016.7493348.

[182] LI, Y., TARLOW, D., BROCKSCHMIDT, M., et al. "Gated graph sequence neural networks", *arXiv:1511.05493*, Sep 2015.

[183] CHO, K., VAN MERRIËNBOER, B., BAHDANAU, D., et al. "On the properties of neural machine translation: Encoder-decoder approaches", *eprint arXiv:1409.1259*, Sep 2014.

[184] "National climactic data center". Disponível em: <`ftp://ftp.ncdc.noaa.gov/pub/data/gsod`>. Acesso em: (2020, Feb. 1).

[185] WINER, B. J. "Statistical principles in experimental design", *McGraw-Hill Book Company*, 1962.

[186] HINTON, G., SRIVASTAVA, N., SWERSKY, K. "Lecture 6e RMSprop: Divide the gradient by a running average of its recent magnitude". 2012.

[187] SHUMAN, D. I., VANDERGHEYNST, P., FROSSARD, P. "Chebyshev polynomial approximation for distributed signal processing". In: *2011 International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS)*, pp. 1–8, June 2011.

[188] TREMBLAY, N., BORGNAT, P. "Subgraph-Based Filterbanks for Graph Signals", *IEEE Transactions on Signal Processing*, v. 64, n. 15, pp. 3827–3840, Mar 2016.

[189] SÜLI, E., MAYERS, D. F. *An introduction to numerical analysis*. Cambridge, Cambridge university press, 2003.

[190] DONOHO, D. L., OTHERS. "Compressed sensing", *IEEE Transactions on information theory*, v. 52, n. 4, pp. 1289–1306, 2006.

[191] KOVAČEVIĆ, J., CHEBIRA, A., OTHERS. "An introduction to frames", *Foundations and Trends in Signal Processing*, v. 2, n. 1, pp. 1–94, 2008.

Apêndice A

Proofs

A.1 Chapter 3

In order to prove Theorem 3.1.2. , we first introduce the following lemmas [101].

Lemma A.1.1. *Let* $w_Q(\lambda) = \sum\limits_{q=0}^{Q} c_q \lambda^q$ *be a* $Q+1$*-order polynomial kernel, then* $d_{\mathcal{G}}(n,m) > Q$ *implies that* $(t_n^{w_Q})_m = 0$*.*

Demonstração. Since $d_{\mathcal{G}}(n,m) > Q$, $(\mathbf{L}^q)_{nm} = 0 \; \forall 0 < q \leq Q$, by property 4 in Section 2.1.2, then:

$$(t_n^{\mathbf{w_Q}})_m = \sqrt{N} \sum_{k=1}^{N} w_Q(\lambda) U_{nk} U_{mk} = \sqrt{N} \sum_{k=1}^{N} \sum_{q=0}^{Q} c_q \lambda^q U_{nk} U_{mk} \tag{A.1}$$

$$\sqrt{N} \sum_{k=1}^{N} \sum_{q=0}^{Q} c_q (\mathbf{L}^Q)_{nm} = 0 \tag{A.2}$$

□

Lemma A.1.2. *([189, Equation (4.6.10)]) If the function* $w(\lambda) : [0, \lambda_{\max}] \rightarrow \mathbb{R}$ *is* $(Q+1)$*-continuously differentiable, then*

$$\inf_{w_q \in \mathcal{P}_Q(\mathbb{R})} \{\|w - w_Q\|_\infty\} \leq \frac{\left(\frac{\lambda_{\max}}{2}\right)^{Q+1}}{(Q+1)!2^Q} \|w^{(Q+1)}\|_\infty, \tag{A.3}$$

where $\mathcal{P}_Q(\mathbb{R})$ *is the space of* Q*-degree polynomials in* $\mathbb{R}$*.*

Lemma A.1.3. *Let* $w : [0, \lambda_{\max}] \rightarrow \mathbb{R}$ *be a spectral kernel and define* $Q_{nm} = d_{\mathcal{G}}(n,m) - 1$*, then*

$$|(t_n^{\mathbf{w}})_m| \leq \sqrt{N} \inf_{\mathcal{P}_{Q_{nm}}(\mathbb{R})} \left\{ \sup_{\lambda \in [0, \lambda_{\max}} |w(\lambda) - w_{Q_{nm}}(\lambda)| \right\} = \sqrt{N} \inf_{\mathbf{w}_{Q_{nm}}} \{\|w - w_{Q_{nm}}\|_\infty\} \tag{A.4}$$

Demonstração. By Lemma A.1.1, $(t_n^{w_{Q_{nm}}})_m = 0$, then:

$$
\begin{aligned}
|(t_n^{\mathbf{w}_{Q_{nm}}})_m| = \inf_{w_{Q_{nm}}} | &= \inf_{w_{Q_{nm}} \in \mathcal{P}_{Q_{nm}}} |(t_n^w)_m - (t_n^{w_{Q_{nm}}})_m| && \text{(A.5)} \\
&= \inf_{w_{Q_{nm}} \in \mathcal{P}_{Q_{nm}}} \left| \sqrt{N} \sum_{k=1}^{N} (w(\lambda) - w_{Q_{nm}}(\lambda)) \bar{u}_{nk} U_{mk} \right| && \text{(A.6)} \\
&\leq \inf_{w_{Q_{nm}} \in \mathcal{P}_{Q_{nm}}} \sqrt{N} \sum_{k=1}^{N} |(w(\lambda) - w_{Q_{nm}}(\lambda)) \bar{u}_{nk} U_{mk}| && \text{(A.7)} \\
&\leq \inf_{w_{Q_{nm}} \in \mathcal{P}_{Q_{nm}}} \sqrt{N} \left[\sum_{k=1}^{N} |(w(\lambda) - w_{Q_{nm}}(\lambda))^i \right]^{1/i} \left[\sum_{k=1}^{N} (\bar{u}_{nk} U_{mk})^j \right]^{1/j}, \quad \frac{1}{j} + \frac{1}{i} = 1 && \text{(A.8)} \\
&\leq \sqrt{N} \inf_{w_{Q_{nm}} \in \mathcal{P}_{Q_{nm}}} \|w - w_{Q_{nm}}\|_i \left[\sum_{k=1}^{N} (\bar{u}_{nk} U_{mk})^j \right]^{1/j} && \text{(A.9)} \\
&\leq \sqrt{N} \inf_{w_{Q_{nm}} \in \mathcal{P}_{Q_{nm}}} \|w - w_{Q_{nm}}\|_i \left[\mu^{2(j-1)} \sum_{k=1}^{N} |\bar{u}_{nk} U_{mk}| \right]^{1/j} && \text{(A.10)} \\
&\leq \sqrt{N} \inf_{w_{Q_{nm}} \in \mathcal{P}_{Q_{nm}}} \|w - w_{Q_{nm}}\|_i \left[\mu^{2(j-1)} \left\{ \sum_{k=1}^{N} |\bar{u}_{nk}|^2 \right\}^{1/2} \left\{ \sum_{k=1}^{N} |U_{mk}|^2 \right\}^{1/2} \right]^{1/j} && \text{(A.11)} \\
&= \sqrt{N} \mu^{\frac{2(j-1)}{i}} \inf_{w_{Q_{nm}} \in \mathcal{P}_{Q_{nm}}} \|w - w_{Q_{nm}}\|_i, \ \frac{1}{j} + \frac{1}{i} = 1, && \text{(A.12)}
\end{aligned}
$$

where (A.7) follows from triangular inequality and (A.8) follows from Holder inequality. Taking $i = j = 2$ leads to the result in (A.4). □

Combining theses these lemmas, we prove Theorem 3.1.2.

Demonstração. Since w is $d_{nm} = d_{\mathcal{G}}(n, m)$ continuously differentiable, substituting the result from LemmaA.1.2 in (A.8):

$$
\begin{aligned}
|(t_n^{\mathbf{w}})_m| &\leq \sqrt{N} \mu \frac{\left(\frac{\lambda_{\max}}{2}\right)^{d_{nm}}}{d_{nm}! 2^{d_{nm}-1}} \|w^{(d_{nm})}\|_\infty && \text{(A.13)} \\
&\leq \frac{2\sqrt{N}}{d_{nm}!} \left(\frac{\lambda_{\max}}{4} \right)^{d_{nm}} \|w^{(d_{nm})}\|_\infty, && \text{(A.14)}
\end{aligned}
$$

and $\|w^{d_{nm}}\|_\infty = \sup_{\lambda \in [0, \lambda_{\max}]} |w^{(d_{nm})}(\lambda)|$. Using Lemma 3.1.1, if $w(0) \neq 0$:

$$
\frac{|(t_n^{\mathbf{w}})_m|}{\|\mathbf{t}_n^{\mathbf{w}}\|_2} \leq \frac{2\sqrt{N}}{|w(0)| d_{nm}!} \left(\frac{\lambda_{\max}}{4} \right)^{d_{nm}} \|w^{(d_{nm})}\|_\infty. \tag{A.15}
$$

□

A.2 Chapter 4

In order to prove Theorem 4.1.1, we first introduce two lemmas from [30].

Lemma A.2.1. *let* $\mathbf{g}_{n,r}$ *and* $\tilde{\mathbf{g}}_{n,r}$ *be the SGWT atoms associated with mother wavelet kernels* g *and* $\tilde{g}$ *respectively, then, if* $|g(\alpha_r\lambda) - \tilde{g}(\alpha_r\lambda)| \leq M(\alpha_r)\ \forall\lambda \in [0, \lambda_{\max}]$*, for each* $n = 1, ..., N$*,* $|(g_{n,r})_m - (\tilde{g}_{n,r})_m| \leq M(\alpha_r)$ *and* $\|g_{n,r} - \tilde{g}_{n,r}\|_2 \leq \sqrt{N}M(\alpha_r)$*.*

Demonstração.

$$|(g_{n,r})_m - (\tilde{g}_{n,r})_m| = \left|\sum_{k=1}^{N} g(\alpha_r\lambda_k)U_{nk}U_{mk} - \sum_{k=1}^{N} \tilde{g}(\alpha_r\lambda_k)U_{nk}U_{mk}\right|. \tag{A.16}$$

$$= \left|\sum_{k=1}^{N}(g(\alpha_r\lambda_k) - \tilde{g}(\alpha_r\lambda_k))U_{nk}U_{mk}\right| \tag{A.17}$$

$$\leq \sum_{k=1}^{N} |g(\alpha_r\lambda_k) - \tilde{g}(\alpha_r\lambda_k)||U_{nk}U_{mk}| \tag{A.18}$$

$$\leq \sum_{k=1}^{N} M(\alpha_r)|U_{nk}U_{mk}| \tag{A.19}$$

$$\leq= M(\alpha_r)\sqrt{(\sum_{k=1}^{N}|U_{nk}|^2)}\sqrt{(\sum_{k=1}^{N}|U_{mk}|^2)} \tag{A.20}$$

$$= M(\alpha_r), \tag{A.21}$$

where (A.20) follows from Cauchy-Schwartz inequality. The second inequality in the Lemma's statements follows from:

$$\|g_{n,r} - \tilde{g}_{n,r}\|_2^2 = \sum_{m=1}^{N} |(g_{n,r})_m - (\tilde{g}_{n,r})_m|^2 \leq NM(\alpha_r)^2. \tag{A.22}$$

□

Lemma A.2.2. *Let* g *be a* $Q+1$ *times continuous differentiable, satiafying g(0)=0,* $g^{(q)}(0) = 0$ *for all* $q < Q$ *and* $g^{Q}(0) = C \neq 0$*. Assuming that there is some* $t' > 0$ *such that* $|g^{(Q+1)}(\lambda)| \leq B$ *for all* $\lambda \in [0, t'\lambda_{\max}]$*, then for* $\tilde{g}(t\lambda)^Q$*:*

$$M(t) = \sup_{\lambda\in[0,\lambda_{\max}} |g(\alpha_r\lambda) - \tilde{g}(\alpha_r\lambda)| \leq t^{Q+1}\frac{\lambda_{\max}^{Q+1}}{(Q+1)!}B, \tag{A.23}$$

for all $t < t'$*.*

Demonstração. Since $g^{(q)}(0) = 0$ for all $q < Q$, the Taylor's expansion with remainder of $g(t\lambda)$ can be expressed as:

$$g(t\lambda) = C\frac{(t\lambda)^Q}{Q!} + g^{(Q+1)}(\lambda^*)\frac{t\lambda)^{Q+1}}{(Q+1)!}, \tag{A.24}$$

for some $\lambda^* \in [0, t\lambda]$. If $t < t'$, then $t\lambda < t'\lambda_{\max}$ for $\lambda \in [0, \lambda_{\max}]$ and $|g^{(Q+1)}(\lambda^*)| \leq B$:

$$|g(\alpha_r\lambda) - \tilde{g}(\alpha_r\lambda)| = \left| g^{(Q+1)}(\lambda^*)\frac{(t\lambda)^{Q+1}}{(Q+1)!} \right| \leq B\frac{t\lambda)^{Q+1}}{(Q+1)!}. \tag{A.25}$$

Since equation (A.25) holds for all $\lambda \in [0, \lambda_{\max}]$, the result in (A.23) follows by taking the sup over $\lambda \in [0, \lambda_{\max}]$. □

Demonstração. (Proof of Theorem 4.1.1) Define $\tilde{g}(\lambda) = \frac{g^{(Q)}(0)}{Q!}\lambda^Q$, then $(\tilde{g}_{n,r})_m = \sum_{k=1}^{N} \frac{g^{(Q)}(0)}{Q!}(\alpha_r\lambda_k)^Q U_{nk}U_{mk} = \sum_{k=1}^{N} \frac{g^{(Q)}(0)}{Q!}\alpha_r^Q L_{nm}^Q = 0$, if $d_{\mathcal{G}}(n,m) > Q$. Then, by Lemma A.2,

$$|(g_{n,r})_m - (\tilde{g}_{n,r})_m| = |(g_{n,r})_m| \leq \alpha_r^{Q+1}C', \; C' = \frac{\lambda_{\max}^{Q+1}}{(Q+1)!}B. \tag{A.26}$$

In order to bound the denominator of $\frac{g_{n,r_m}}{\|g_{n,r}\|_2}$, note that $g_{n,r} = \tilde{g}_{n,r} + (g_{n,r} - \tilde{g}_{n,r})$, and by triangular inequality,

$$\|g_{n,r}\|_2 \geq \|\tilde{g}_{n,r}\|_2 - \|g_{n,r} - \tilde{g}_{n,r}\|_2. \tag{A.27}$$

Since $\|\tilde{g}_{n,r}\|_2 = \|\frac{g^{(Q)}(0)}{Q!}\alpha_r^Q(\mathbf{L}^Q)\boldsymbol{\delta}_n\|_2$ and

$$\|g_{n,r} - \tilde{g}_{n,r}\|_2 \leq \sqrt{N}\alpha_r^{Q+1}\frac{\lambda_{\max}^{Q+1}}{(Q+1)!}B, \tag{A.28}$$

by Lemma A.2.2, which implies that

$$\|g_{n,r}\|_2 \geq \sqrt{N}\alpha_r^Q\left(\frac{g^{(Q)}(0)}{Q!}\|\mathbf{L}^Q\boldsymbol{\delta}_n\|_2 - \alpha_r\frac{\lambda_{\max}^{Q+1}}{(Q+1)!}B\right), \tag{A.29}$$

and combining with the previous result, it follows that

$$\frac{(g_{n,r})_m}{\|g_{n,r}\|_2} \leq \frac{\alpha_r C'}{a - \alpha_r b}, \tag{A.30}$$

with $a = \frac{g^{(Q)}(0)}{Q!}\|\mathbf{L}^Q\boldsymbol{\delta}_n\|_2$ and $b = \alpha_r\frac{\lambda_{\max}^{Q+1}}{(Q+1)!}B$. Defining $r'' = \frac{g^{(Q)}(0)\|\mathbf{L}^Q\boldsymbol{\delta}_n\|_2(Q+1)}{2\sqrt{N}\lambda_{\max}^{Q+1}B}$ and $D = \frac{2C'Q!}{g^{(Q)}(0)\|\mathbf{L}^Q\boldsymbol{\delta}_n\|_2}$, and taking $\alpha_r \leq \frac{a}{2b}$, then, from the following computation follows the desired result.

$$\alpha_r \leq \frac{a}{2b} \iff a - b\alpha_r \geq \frac{a}{2} \iff \frac{\alpha_r C'}{a - \alpha_r b} \leq \frac{2C'}{a}\alpha_r.$$

□

Apêndice B

Random Sampling of Bandlimited GS

A common concern in extending classical DSP methods to GSP is that, depending on the application, graphs can become very large and the GFT, a fundamental component of GSP, is computed globally, being not scalable. Therefore, the aforementioned sampling method, that requires the computation of at least K Laplacian eigenvectors, becomes prohibitive as the size of the graph increases. To deal with this computational cost problem, inspired by *compressed sensing*, a different sampling strategy based on random sampling was proposed in [117]. In Theorem 5.1.4, the minimum acceptable sampling size is $|\mathcal{F}| = K$, but in random sampling, slightly more samples $M > K$ are kept in-sample.

Remark B.0.1. *Although [117] considers $\mathcal{F} = \{1, 2, .., K\}$, where the Laplacian eigenvectors are generally sorted in an ascending ordering of eigenvalues, most of the results holds for arbitrary $\mathcal{F}$. Because error bounding results for the interpolation formula* (B.8) *require monotonicity of the graph polynomial $h : [0, \lambda_{\max}] \to \mathbb{R}$, this section considers $\mathcal{F} = \{m, m+1, ..., m+K-1\}$, $m < N - K$ (pass-band).*

B.1 Sample Size Bounds

Definition B.1.1 describes a sampling operator similar to (5.2) but with random entries given by a priori distribution instead. Thereafter, Theorem B.1.3 provides a minimum number of samples M such that, with high probability, the original signal can be recovered with a small error.

Definition B.1.1. *(Random sampling operator) Let $\mathbf{p} \in \mathbb{R}^N$ be a vector containing a probability distribution on $\mathcal{V}$ ($\sum_n p_n = 1$, $p_n \geq 0\ \forall n$) and $\widetilde{\mathcal{S}} = \{\Omega_1, ..., \Omega_M\}$ be a subset of nodes drawn from $\mathcal{V}$ with replacement such that $\mathbb{P}\{\Omega_m = n\} = p_n$. The* *random sampling matrix* $\widetilde{\mathbf{\Psi}}_{\mathcal{S}}$ *with respect to the distribution* $\mathbf{p}$ *is given by*

$$(\widetilde{\mathbf{\Psi}}_{\mathcal{S}})_{mn} = \begin{cases} 1, & \Omega_m = n \\ 0 & \text{elsewhere.} \end{cases} \tag{B.1}$$

Definition B.1.2. *(Graph weighted coherence)*

Let $\mathbf{p}$ *be a vector containing a probability distribution on* $\mathcal{V}$*. The graph weighted coherence* $\nu_{\mathbf{p}}^{\mathcal{F}}$ *is defined as*

$$\nu_{\mathbf{p}}^{\mathcal{F}} \triangleq \max_{n}\{p_n^{-1/2}\|\mathbf{U}_{:,\mathcal{F}}\boldsymbol{\delta}_n\|_2\} \tag{B.2}$$

Remember that the graph coherence μ introduced in Lemma 3.1.1 is the maximum absolute value in the GFT matrix $\mu = \max_{n,k} U_{nk}$. The graph weighted coherence (B.2) is equivalent to the graph coherence associated with the submatrix $\mathbf{U}_{\mathcal{F}}$ with nodes weighted by the probability distribution $\mathbf{p}$. The local coherence $\|\mathbf{U}_{:,\mathcal{F}}\boldsymbol{\delta}_n\|_2$ measures how much energy of GSs in $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$ is concentrated on node n. Since the columns of $\mathbf{U}$ are unit norm, the local coherence is bounded by zero and 1. If the local coherence is close to zero, then node n does not keep much information of signals in $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$ and can be dropped. If the local coherence is close to 1, on the other hand, node n is informative about $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$ and should be selected by the sampling operator. The distribution $\mathbf{p}$ weights the importance of each node based on prior information.

Theorem B.1.3. *Let* $\widetilde{\mathbf{\Psi}}_{\mathcal{S}}$ *be a random sampling operator as in* (B.1.1) *and* $\mathbf{P}$ *be a diagonal matrix with* $[p_{\Omega_1}, \ldots, p_{\Omega_M}]$ *in the diagonal. Given any* $\delta, \epsilon \in (0, 1)$*, with probability* $1 - \epsilon$

$$(1-\delta)\|\mathbf{x}\|_2^2 \leq \frac{1}{M}\|\widetilde{\mathbf{\Psi}}_{\mathcal{S}}\mathbf{P}^{-1/2}\mathbf{x}\|_2^2 \leq (1+\delta)\|\mathbf{x}\|_2^2 \tag{B.3}$$

for all $\mathbf{x} \in \mathrm{BL}_{:,\mathcal{F}}(\mathbf{U})$ *provided*

$$M \geq \frac{3}{\delta}(\nu_{\mathbf{p}}^{\mathcal{F}})^2 \log\left(\frac{K}{\epsilon}\right) \tag{B.4}$$

Theorem B.1.3 shows that $\frac{1}{M}\widetilde{\mathbf{\Psi}}_{\mathcal{S}}\mathbf{P}^{-1/2}$ embeds the $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$ into $\mathbb{R}^M$. The main difference between this result and *compressive sensing* [190] is the knowledge of the GS support $\mathcal{F}$ in addition to the support size K. Note that $\sqrt{K}$ is a lower bound for $\nu_{\mathbf{p}}^{\mathcal{F}}$:

$$(\nu_{\mathbf{p}}^{\mathcal{F}})^2 = \max_{n}\{p_n^{-1/2}\|\mathbf{U}_{:,\mathcal{F}}\boldsymbol{\delta}_n\|_2\} \geq \sum_n \frac{\|\mathbf{U}_{:,\mathcal{F}}\boldsymbol{\delta}_n\|_2}{p_n}p_n = \sum_n \|\mathbf{U}_{:,\mathcal{F}}\boldsymbol{\delta}_n\|_2 \geq K,$$

then, for given ϵ and δ, the optimal distribution $p_n^* = \frac{\|\mathbf{U}_{\mathcal{F}}^{\mathrm{T}}\boldsymbol{\delta}_n\|_2^2}{K}$ leads the smallest lower bound (B.4) in Theorem B.1.3:

$$M \geq \frac{3}{\delta} K \log\left(\frac{K}{\epsilon}\right) \tag{B.5}$$

Note that $\mathbf{p}^*$ depends on $\mathbf{U}_{\mathcal{F}}$, which we wanted to avoid to compute, hence, an approximation for $\mathbf{p}^*$ is proposed in [117]. The goal of this section is to give a brief overview of random sampling in GSP, further details and proofs can be found in [117].

B.2 Recovery of Randomly Sampled GSs

Let $\mathbf{y} = \widetilde{\boldsymbol{\Psi}}_{\mathcal{S}}\mathbf{x} + \boldsymbol{\eta}$ be a sampled GS with uncorrelated zero mean Gaussian noise $\boldsymbol{\eta}$, then the original signal $\mathbf{x} \in \mathrm{BL}_{\mathcal{F}}(\mathbf{U})$ can be approximated by the following optimization problem:

$$\tilde{\mathbf{x}} = \min_{\mathbf{z}\in\mathrm{BL}_{\mathcal{F}}(\mathbf{U})} \left\|\mathbf{P}_{\widetilde{\mathcal{S}}}^{-1/2}(\mathbf{y} - \widetilde{\boldsymbol{\Psi}}_{\mathcal{S}}\mathbf{z})\right\|_2. \tag{B.6}$$

Note that recovering $\mathbf{x}$ by (B.6) requires at least a basis for $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$. In order to approximate the original GS $\mathbf{x}$ from the sampled GS $\mathbf{y}$, consider the bandpass polynomial filter $h : [0, \lambda_{\max}] \to \mathbb{R}$ with passband $\mathcal{F}$:

$$\tilde{\mathbf{x}} = \min_{\mathbf{z}\in\mathbb{R}^N} \left\|\mathbf{P}_{\widetilde{\mathcal{S}}}^{-1/2}(\mathbf{y} - \widetilde{\boldsymbol{\Psi}}_{\mathcal{S}}\mathbf{z})\right\|_2^2 + \gamma\mathbf{z}^{\mathrm{T}}(\mathbf{I}_N - h(\mathbf{L}))\mathbf{z}, \gamma > 0, \tag{B.7}$$

with explicit solution:

$$\left(\widetilde{\boldsymbol{\Psi}}_{\mathcal{S}}^{\mathrm{T}}\mathbf{P}_{\widetilde{\mathcal{S}}}^{-1}\widetilde{\boldsymbol{\Psi}}_{\mathcal{S}} + \gamma(\mathbf{I}_N - h(\mathbf{L}))\right)\tilde{\mathbf{x}} = \widetilde{\boldsymbol{\Psi}}_{\mathcal{S}}^{\mathrm{T}}\mathbf{P}_{\widetilde{\mathcal{S}}}^{-1}\mathbf{y} \tag{B.8}$$

Note that the regularization term $\mathbf{z}^{\mathrm{T}}(\mathbf{I}_N - h(\mathbf{L}))\mathbf{z}$ in (B.7) penalizes non-$\mathcal{F}$-bandlimited GS and is a relaxation of the set constraint in (B.6). Since [117] considers $\mathcal{F} = \{1, 2, .., K\}$, the polynomial h is required to be a high pass filter. Nonetheless, as mentioned by Remark B.0.1, this section considers $\mathcal{F}$ as a pass-band, thus the highpass polynomial filter in [117, Equation 9] is replaced by the reject-band polynomial filter $1 - h(\lambda)$ in equation (B.7). It is worth mentioning that, $\mathcal{F} = \{1, 2, .., K\}$ is a common assumption for GSs, since it is associated with smoothness.

For $\mathcal{F} = \{1, 2, .., K\}$ and h a non-decreasing and non-negative polynomial with $h(\lambda_{K+1}) > 0$, Theorem 7 in [117] provides upper bounds for

- $\|\mathbf{U}_{:,\mathcal{F}}\mathbf{U}_{:,\mathcal{F}}^{\mathrm{T}}\tilde{\mathbf{x}} - \mathbf{x}\|_2$, the Euclidean distance between the original GS $\mathbf{x}$ and the

GS recovered by (B.8) and projected to $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$;

- $\|(\mathbf{I}_N - \mathbf{U}_{:,\mathcal{F}}\mathbf{U}_{:,\mathcal{F}}^{\mathrm{T}})\tilde{\mathbf{x}}\|_2$, the norm of the components of the solution $\tilde{\mathbf{x}}$ orthogonal to $\mathrm{BL}_{\mathcal{F}}(\mathbf{U})$.

In [117, Theorem 7], the graph polynomial h is requires to be non-decreasing and non-negative highpass filter, this filter can be implemented by the Jackson-Chebyshev approximation of an ideal high-pass filter.

Apêndice C

Frames on Graphs

C.1 WGFT

In DSP, tight frames allow the interpretation of spectograms as an energy density function, improve stability in recovering signals from noisy measurements, and also provide faster computations [104]. A frame is a generalization of a basis from linear algebra that allows linearly dependent elements. This characteristic provides redundant representations which can allow sparse, simple, and/or robust representations of a signal [191]. A redundant frame is also called overcomplete representation. Before introducing the frame analysis of the WGFT, we take a brief review on frame theory.

Consider the finite[1] N-dimensional vector space $\mathbb{V}$, the set of vectors $\{\mathbf{v}_m\}_{m\in\mathbb{N}} \in \mathbb{V}$ is a frame if there exist positive constants, also called frame bounds, $0 < A \leq B < \infty$ such that, $\forall \mathbf{x} \in \mathbb{V}$, the following inequalities hold:

$$A\|\mathbf{x}\|^2 \leq \sum_{m\in\mathbb{N}} |\langle \mathbf{v}_m, \mathbf{x}\rangle|^2 \leq B\|\mathbf{x}\|^2\,. \tag{C.1}$$

The inequalities in (C.1) can be seen as a generalization of Parseval's identity in (2.19). For instance, a basis is a frame which is not redundant, that is, removing elements from $\{\mathbf{v}_n\}_{n\in\mathbb{N}}$ impairs the spanning of the vector space $\mathbb{V}$ and an orthonormal basis is a frame with constants $A = B = 1$.[2] When $A = B$ the frame is said to be tight, moreover, an A-tight frame $\{\mathbf{v}_m\}_{m=1}^{M}$, $M > N$ for a finite N-dimensional vector space $\mathbb{V}^N$ satisfies

$$A\mathbf{x} = \sum_{m=1}^{M} \langle \mathbf{x}, \mathbf{v}_m\rangle \mathbf{v}_m. \tag{C.2}$$

[1] Although frame theory is applicable to inner dot vector spaces, this text will limit to finite vector spaces since we are only dealing with finite graphs

[2] The reverse is not true: a frame with frame bounds $A = B = 1$ is not necessarily a basis.

Tight frames tend to provide stable representations, as shown in the following example:

Example C.1.1. *Consider the tight frame* $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\} \subset \mathbb{R}^2$*,* $\|\mathbf{v}_m\|_2 = 1$*, and let* $\mathbf{x} \in \mathbb{R}^2$ *be a vector such that*

$$\mathbf{x} = \frac{1}{A}\sum_{m=1}^{3}\langle \mathbf{x}, \mathbf{v}_m\rangle \mathbf{v}_m \,. \tag{C.3}$$

The frame $\{\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3\}$ *is redundant since removing one of its vector elements may still span* $\mathbb{R}^2$*. Also, suppose that the frame coefficients are perturbed by* $\boldsymbol{\epsilon} = [\epsilon_1, \epsilon_2, \epsilon_3]^{\mathrm{T}}$*, then the recovered* $\mathbf{x}$*,* $\tilde{\mathbf{x}}$*, is perturbed by* $\frac{1}{A}\sum\limits_{m=1}^{3}\epsilon_m \mathbf{v}_m$ *and the squared error is bounded by the squared norm of the perturbation vector as follows:*

$$\|\tilde{\mathbf{x}} - \mathbf{x}\|_2^2 = \|\sum_{m=1}^{3}\epsilon_m \mathbf{v}_m\|_2^2 \tag{C.4}$$

$$\leq \frac{1}{A}\sum_{m=1}^{3}\epsilon_m^2\|\mathbf{v}_m\|_2^2 = \frac{1}{A}\|\epsilon\|_2^2. \tag{C.5}$$

Thus, a small perturbation of the of the frame coefficients leads to small errors.

It is worth pointing out that the dictionary of WGFT atoms $\{\mathbf{w}_{n,k}\}_{n,k=1}^{N}$ rarely defines a tight frame, only if $\mu = \frac{1}{\sqrt{N}}$. The following theorem gives the frame bounds for the frame composed by the WGFT atoms.

Theorem C.1.2 ([101], Theorem 3)**.** *If* $w(0) \neq 0$*, then* $\{\mathbf{w}_{n,k}\}_{n,k=1}^{N}$ *is a frame with constants* A *and* B *defined as*

$$0 < |w(0)|^2 \leq A \triangleq \min_n\{\|\mathbf{t}_n^w\|_2^2\} \leq$$
$$B \triangleq \max_n\{\|\mathbf{t}_n^w\|_2^2\} \leq \mu^2\|\mathbf{w}\|_2^2$$

Demonstração.

$$\sum_{n=1}^{N}\sum_{k=1}^{N}|\mathbf{x}^{\mathrm{T}}\mathbf{w}_{n,k}|^2 = \sum_{n=1}^{N}\sum_{k=1}^{N}|\mathbf{x}^{\mathrm{T}}\left[(\mathbf{U}\mathbf{W}\bar{\mathbf{u}}_n)\odot \mathbf{u}_k\right]|^2$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{N}|\mathbf{x}^{\mathrm{T}}\odot(\mathbf{U}\mathbf{W}\bar{\mathbf{u}}_n)^{\mathrm{T}}\mathbf{u}_k|^2 \tag{C.6}$$

$$= \sum_{n=1}^{N}\sum_{k=1}^{N}(\mathbf{x}\widehat{\odot\,\mathbf{U}\mathbf{W}}\bar{\mathbf{u}}_n)_k^2 = \sum_{n=1}^{N}\sum_{m=1}^{N}(\mathbf{x}\odot\mathbf{U}\mathbf{W}\bar{\mathbf{u}}_n))_m^2 \tag{C.7}$$

$$= \sum_{n=1}^{N}\sum_{m=1}^{N}|x_m|^2|(\mathbf{U}\mathbf{W}\bar{\mathbf{u}}_n)_m|^2 = \sum_{m=1}^{N}|x_m|^2\|(\mathbf{t}_m^w)_n\|_2^2 \tag{C.8}$$

where the property of inner product with kron product was used in (C.6), the Parseval's identity was used in (C.7), and the Laplacian symmetry, in (C.8).

Moreover, if $w(0) \neq 0$, then $0 < |w(0)|^2 \leq \sum_{n=1}^{N} |w(\lambda_k)|^2 |U_{nk}|^2 = \|\mathbf{t}_n^{\mathbf{w}}\|_2^2$.

□

C.2 SGWT

Proof of Theorem 4.1.3 [30].

Demonstração. For each $r \in \{1, \dots, R\}$, we have that

$$\begin{aligned}
\sum_n |x_{n,r}^g|^2 &= \sum_{n=1}^{N} \left(\sum_{k=1}^{N} g_r(\lambda_k)\hat{x}_k U_{nk} \sum_{\ell=1}^{N} g_r(\lambda_\ell)\hat{x}_k U_{n\ell} \right) && \text{(C.9)}\\
&= \sum_{\ell=1}^{N} \sum_{k=1}^{N} g_r(\lambda_k)\hat{x}_k g_r(\lambda_\ell)\hat{x}_\ell \sum_{n=1}^{N} U_{nk} U_{n\ell} && \text{(C.10)}\\
&= \sum_{k=1}^{N} g_r^2(\lambda_k)\hat{x}_k^2 && \text{(C.11)}
\end{aligned}$$

Similarly $\sum_{n=1}^{N} |x_n^h|^2 = \sum_{k=1}^{N} h^2(\lambda_k)\hat{x}_k^2$. Thus

$$\begin{aligned}
\sum_{n=1}^{N} |x_n^h|^2 + \sum_{r=1}^{R} \sum_n |x_{n,r}^g|^2 &= \sum_{k=1}^{N} h^2(\lambda_k)\hat{x}_k^2 + \sum_{r=1}^{R} \sum_{k=1}^{N} g_r^2(\lambda_k)\hat{x}_k^2\\
&= \sum_{k=1}^{N} \left(h^2(\lambda_k) + \sum_{r=1}^{R} g_r^2(\lambda_k) \right) \hat{x}_k^2\\
&= \sum_{k=1}^{N} G(\lambda)\hat{x}_k^2 \leq \max_{\lambda \in [0, \lambda_{\max}]} G(\lambda) \|\mathbf{x}\|_2^2 && \text{(C.12)}
\end{aligned}$$

where the inequality in (C.12) holds by Parseval's identity. Similarly, $\sum_{n=1}^{N} |x_n^h|^2 + \sum_{r=1}^{R} \sum_n |x_{n,r}^g|^2 \geq \max_{\lambda \in [0, \lambda_{\max}]} G(\lambda) \|\mathbf{x}\|_2^2$ □