ESTIMATION OF PARTIALLY OBSERVED MULTIPLE GRAPH SIGNALS BY LEARNING SPECTRALLY CONCENTRATED GRAPH KERNELS

A THESIS SUBMITTED TO
THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES
OF
MIDDLE EAST TECHNICAL UNIVERSITY

BY

GÜLCE TURHAN

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR
THE DEGREE OF MASTER OF SCIENCE
IN
ELECTRICAL AND ELECTRONICS ENGINEERING

FEBRUARY 2021

Approval of the thesis:

ESTIMATION OF PARTIALLY OBSERVED MULTIPLE GRAPH SIGNALS BY LEARNING SPECTRALLY CONCENTRATED GRAPH KERNELS

submitted by **GÜLCE TURHAN** in partial fulfillment of the requirements for the degree of **Master of Science in Electrical and Electronics Engineering Department, Middle East Technical University** by,

Prof. Dr. Halil Kalıpçılar
Dean, Graduate School of **Natural and Applied Sciences** _____

Prof. Dr. İlkay Ulusoy
Head of Department, **Electrical and Electronics Engineering** _____

Assoc. Prof. Dr. Elif Vural
Supervisor, **Electrical and Electronics Engineering, METU** _____

Examining Committee Members:

Prof. Dr. Gözde Bozdağı Akar
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Elif Vural
Electrical and Electronics Engineering, METU _____

Assoc. Prof. Dr. Sevinç Figen Öktem
Electrical and Electronics Engineering, METU _____

Assist. Prof. Dr. Mustafa Mert Ankaralı
Electrical and Electronics Engineering, METU _____

Prof. Dr. Osman Parlaktuna
Electrical and Electronics Engineering, OGU _____

Date:

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Surname: Gülce Turhan

Signature :

ABSTRACT

ESTIMATION OF PARTIALLY OBSERVED MULTIPLE GRAPH SIGNALS BY LEARNING SPECTRALLY CONCENTRATED GRAPH KERNELS

Turhan, Gülce
M.S., Department of Electrical and Electronics Engineering
Supervisor: Assoc. Prof. Dr. Elif Vural

February 2021, 69 pages

Graph models provide flexible tools for the representation and analysis of signals defined over domains such as social or sensor networks. However, in real applications data observations are often not available over the whole graph, due to practical problems such as broken sensors, connection loss, or storage problems. In this thesis, we study the problem of estimating partially observed graph signals on multiple graphs. We consider possibly multiple graph domains over which a set of signals is available with missing observations. We study the problem of learning a graph signal model that allows an accurate estimation of the missing observations. The proposed method is based on learning a sparse representation of the graph signals over spectrally characterized graph dictionaries. The dictionary on each graph consists of a set of spectrally concentrated, narrowband graph atoms localized at different graph nodes. We formulate the dictionary learning problem in the spectral domain, as opposed to the vertex domain, which provides the flexibility of incorporating signals from more than one graph in the learning. The learnt dictionaries consist of several sub-dictionaries, where each sub-dictionary consists of atoms with a spectrum concentrated at a certain graph frequency, so that each sub-dictionary captures a different

spectral component of the graph signals at hand. We approximate the narrowband graph spectra with Gaussian kernels, the parameters of which are learnt jointly with the sparse coefficients of the graph signals. The resulting optimization problem is solved with an alternating optimization approach. Finally, the incomplete entries of the given graph signals are estimated using the learnt dictionaries and sparse coefficients. Experimental results on synthetic graph data sets suggest that the proposed method has promising performance in comparison to baseline solutions.

Keywords: Sparse representation, graph signal processing, dictionary learning, graph kernels, multiple graph domains, optimization

ÖZ

KISMEN GÖZLENEN ÇOKLU GRAF SİNYALLERİNİN DAR BANTLI GRAF KERNELLERİ ÖĞRENİLEREK KESTİRİMİ

Turhan, Gülce
Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü
Tez Yöneticisi: Doç. Dr. Elif Vural

Şubat 2021 , 69 sayfa

Graf modelleri, sosyal ağlar, sensör ağları veya ulaşım ağları üzerindeki uygulamalarda sinyallerin gösterimi ve analizinde yaygın olarak kullanılmaktadır. Buna karşılık, gerçek uygulamalarda elde edilen verilerde sensör bozulması, bağlantı kaybı, kısıtlı depolama imkanları gibi bazı problemlerden dolayı graf sinyalinin tüm graf üzerindeki değerleri eksiksiz olarak gözlemlenemeyebilir. Bu tezde, çoklu graflar üzerinde kısmen gözlemlenen graf sinyallerinin kestirimi problemi incelenmiştir. Birden çok graf üzerinde, sinyallerin bazı girdilerinin eksik olarak gözlemlendiği bir durum ele alınmış; eksik gözlemlerin doğru bir şekilde kestirimini sağlayan bir graf sinyal modeli öğrenme problemi incelenmiştir. Önerilen yöntem, spektral olarak karakterize edilmiş graf sözlükleri üzerinden graf sinyallerinin seyrek bir temsilini öğrenmeyi amaçlamıştır. Her graf sözlüğü, farklı graf nodlarında lokalize olmuş bir dizi dar bant graf atomundan oluşmaktadır. Sözlük öğrenme problemi nod alanının aksine, birden fazla graftan elde edilen sinyal verilerini birleştirme esnekliği sağlayan spektral alanda formüle edilmiştir. Öğrenilen sözlükler, belirli bir graf frekansında merkezlenmiş dar bant spektrumlu atomlardan oluşan birkaç alt sözlükten oluşmakta

ve böylece her bir alt sözlük, eldeki graf sinyallerinin farklı bir spektral bileşenini ifade etmeye yaramaktadır. Dar bant graf spektrumları Gauss kernelleri ile ifade edilmiş; Gauss kernel parametreleri ile graf sinyallerinin seyrek gösterim katsayıları ortak olarak öğrenilmiştir. Ortaya çıkan optimizasyon problemi, alternatif optimizasyon yaklaşımıyla çözülmüştür. Sonuç olarak, öğrenilen sözlükler ve seyrek katsayılar yardımıyla verilen graf sinyallerinin eksik gözlemleri kestirilmiştir. Önerilen methodun sentetik graf veri setlerine ilişkin deney sonuçları, temel çözümlere kıyasla umut verici bir performansa sahiptir.

Anahtar Kelimeler: Seyrek gösterim (temsil), graf sinyal işleme, sözlük öğrenme, graf kernelleri, çoklu graf alanları, optimizasyon

To my family!

ACKNOWLEDGMENTS

Firstly, I would like to thank my thesis supervisor, Assoc. Prof. Dr. Elif VURAL, for her guidance, understanding and consideration. Her broad vision and motivation facilitated the progress of my studies immensely, and I regard it as a privilege to work with her.

Special thanks to my husband Hasan İhsan TURHAN and my daughter Begüm TURHAN, they didn't lose their faith in me, encourage me all the time.

Finally, I would like to thank my family and for their love and support.

TABLE OF CONTENTS

ABSTRACT v

ÖZ vii

ACKNOWLEDGMENTS x

TABLE OF CONTENTS xi

LIST OF TABLES xiv

LIST OF FIGURES xv

LIST OF ALGORITHMS xvii

CHAPTERS

1 INTRODUCTION 1

1.1 Motivation and Problem Definition 1

1.2 The Outline of the Thesis 6

2 RELATED WORK 7

2.1 Graph Based Semi Supervised Learning Algorithms 7

2.2 Signal Processing on Graphs 10

2.2.1 Weighted Graph and Graph Signals 12

2.2.2 Graph Laplacian 13

2.2.3 Graph Fourier Transform 15

2.2.4 Filtering on Graphs 16

2.2.5 Convolution on Graphs . . . 17

2.2.6 Translation on Graphs . . . 17

2.3 Dictionary Learning . . . 17

2.3.1 Dictionary Learning for Signal Approximation and Classification in Regular Domains . . . 19

2.3.2 Domain Adaptive Dictionary Learning in Regular Domains . . . 21

2.3.3 Dictionary Learning on Graph Domains . . . 23

2.3.4 Dictionary Learning in Regular Domains that Model Data with Graphs . . . 27

3 PROPOSED METHOD FOR ESTIMATING PARTIALLY OBSERVED GRAPH SIGNALS . . . 29

3.1 Graph Dictionary Model . . . 30

3.2 Graph Signal Model . . . 32

3.3 Proposed Method and Objective Function . . . 34

3.4 Complexity Analysis of the Algorithm . . . 41

4 EXPERIMENTAL RESULTS . . . 45

4.1 Synthetic Data Generation . . . 45

4.2 Experimental Results . . . 48

4.2.1 Comparative Evaluation of Algorithm Performance . . . 49

4.2.2 Learning the Dictionary with Different Number of Graph Kernels . . . 53

4.2.3 The Effect of the Sparsity Constraints on the Performance . . . 54

4.2.4 Comparison of Dictionary Learning on Single and Multiple Graphs . . . 55

4.2.5 Algorithm Parameters Sensitivity Analysis . . . 56

5 CONCLUSION 59

REFERENCES 63

LIST OF TABLES

TABLES

Table 4.1 Comparison of the MSE values for dictionary learning on multiple graphs (MG) and a single graph (SG) 56

Table 4.2 Comparison of the MSE values for different values of η_s for $\mathcal{G}_1$ and $\mathcal{G}_2$ 57

Table 4.3 Comparison of the MSE values for different values of the η_w-η_x pair. 57

LIST OF FIGURES

FIGURES

Figure 1.1 A partially observed graph signal. The missing entries are illustrated with question marks. 2

Figure 1.2 At the left side of the figure, a graph signal which belongs to a real wind speed data set of Molene, France is illustrated. At the right side, its representation in the graph spectral domain is given. One can observe that the energy of the signal is rather concentrated at rather high frequencies. 3

Figure 1.3 At the left side of the figure, a low-pass Gaussian kernel and two different band-pass Gaussian kernels are illustrated. Respectively, (a) is a low-pass dictionary atom and (b), (c) are band-pass dictionary atoms generated from the kernels shown next to them. One can observe that as the spectrum of the atom shifts to high frequencies, its variation over the graph becomes more rapid. 5

Figure 2.1 (a) Graph structure for a temperature sensor network, (b) a graph signal whose entries are temperature measurements in *Kelvin* taken at a certain date and hour. 10

Figure 2.2 Illustration of some graph construction methods 11

Figure 2.3 A graph signal from a temperature dataset 13

Figure 2.4 Illustration of nodes, edges and degrees 14

Figure 2.5 Three eigenvectors of the graph Laplacian of a random sensor network graph. The eigenvector u_0 is constant at each vertex, the eigenvector u_1 varies slowly across the graph whereas u_{100} contains many zero crossings. 16

Figure 3.1 Three atoms of a subdictionary are illustrated. These atoms are generated from the same low-pass graph kernel by localizing it on different graph nodes. The graph nodes are shown with black boxes. 31

Figure 3.2 Three different sub-dictionary atoms which are localized on the same node n are illustrated. The sub-dictionaries generated from (a) a low-pass kernel and (b), (c) two different band-pass kernels. 32

Figure 4.1 Generated graphs $\mathcal{G}_1$ (a) and $\mathcal{G}_2$ (b) for the simulations. 46

Figure 4.2 Illustration of the Gaussian kernels generated with the ψ_{syn} vector as a function of the eigenvalues of the two synthetic graphs 47

Figure 4.3 MSE values for different rates of missing measurements when LASSO is used in the sparse coding stage. 51

Figure 4.4 MSE values for different rates of missing measurements when OMP is used in the sparse coding stage. 51

Figure 4.5 MSE values for different SNR (dB) values when LASSO is used in the sparse coding stage. 52

Figure 4.6 MSE values for different SNR (dB) values when OMP is used in the sparse coding stage. 53

Figure 4.7 Comparison of the MSE values for different number of Gaussian kernels used in the dictionaries. 54

Figure 4.8 Comparison of the MSE values for different sparsity values T_0 used in the sparse coding stage. 55

LIST OF ALGORITHMS

ALGORITHMS

Algorithm 1 Spectrally Concentrated Graph Dictionary Learning (SCGDL) . 41

CHAPTER 1

INTRODUCTION

1.1 Motivation and Problem Definition

In recent years, there has been growing interest in graph signal processing. Graphs are useful tools to handle datasets where the underlying structure carries important information for interpreting data measurements such as in energy, transportation, social and sensor networks. These networks can be represented with a graph whose nodes represent entities such as sensors or users where the edges between nodes model the interactions between them. A graph signal is a real function defined on the set of graph nodes. In many applications, data samples observed on a graph can be modelled as a graph signal. For instance, temperature measurements taken over a sensor network can be regarded as a signal on the graph that represents the network.

In real world applications, data observations are often not available over the whole graph, typically due to practical problems such as broken sensors, connection loss, or storage problems. An example of a graph signal with missing observations is illustrated in Figure 1.1. The estimation of the missing observations of a partially observed graph signal is a problem that is relevant for a variety of applications. For example in social networks, the interests of social media users, such as favorite books, magazines or movies may not be available on their profiles. The determination of the interests of users can be important in many applications such as recommender systems, advertisements, etc. At this point graph models provide efficient tools for inferring the unknown interests of users from the known ones using the interactions between them.

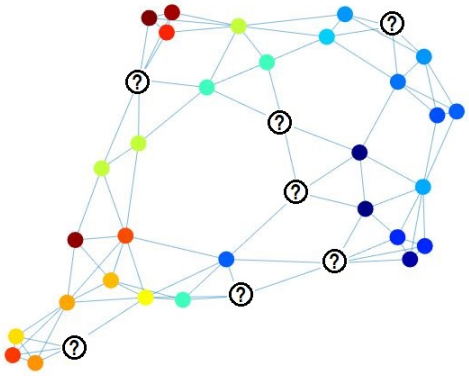


Figure 1.1: A partially observed graph signal. The missing entries are illustrated with question marks.

In the recent years, there has been important developments towards the representation and processing of graph signals. The main challenge in graph signal processing is that graphs have irregular structures, which makes it difficult to perform even very simple tasks, such as the translation of a signal on the graph. Several important works in the recent years have focused on the extension of classical concepts such as Fourier transform, convolution and filtering to graph domains [1], [2]. Due to these results, just like the regular signals in traditional signal processing, graph signals can also be represented in two domains: the vertex domain and the graph spectral domain, i.e., the frequency domain [3].

A common assumption in many machine learning methods is that graph signals change smoothly on the graph and therefore, their energy is mostly concentrated at the low-frequency part of the spectrum. However, in many applications, the frequency content of graph signals can concantrate on different parts of the spectrum as illustrated in Figure 1.2. This makes it hard to estimate the missing observations.

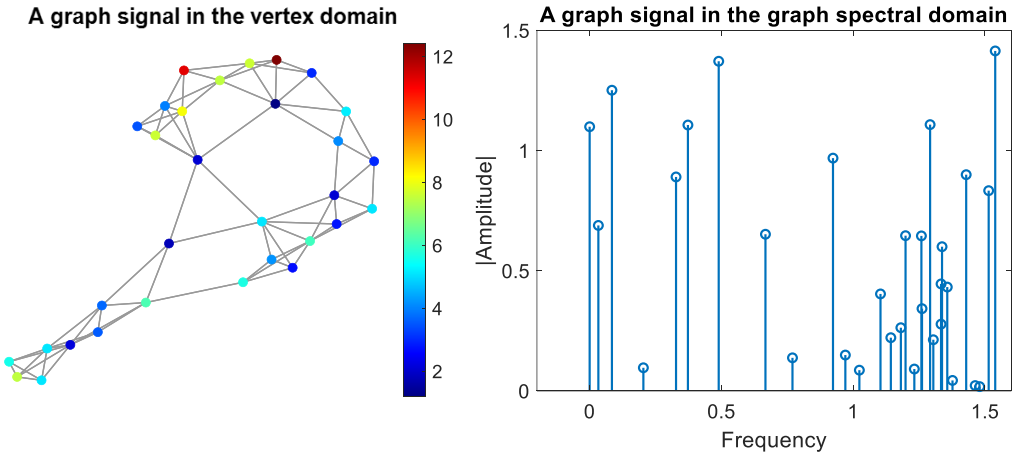


Figure 1.2: At the left side of the figure, a graph signal which belongs to a real wind speed data set of Molene, France is illustrated. At the right side, its representation in the graph spectral domain is given. One can observe that the energy of the signal is rather concentrated at rather high frequencies.

For instance, consider a graph signal that represents some meteorological measurements, such as temperature measurements, over a sensor network in a country. The measurements will have certain global characteristics (e.g., cold weather in winter) called macroclima, that will vary slowly over the whole country. However, in some small isolated regions like lakeside or mountainous regions, the climate may vary locally and rapidly, which is called microclima. These microclimate regions will cause high frequency components in the spectrum of the graph signal that represents the meteorological measurements over the whole country. Similarly, in a social network, a small and isolated group of users may share interests that may be different from the larger community they live in. In order to analyze and model this kind of local variations in an effective way, the used signal models must be able to capture different parts of the spectrum successfully, in a manner that is adapted to the particular structure of the graph signals at hand.

In this study, we propose to model graph signals using graph dictionaries whose atoms are concentrated at a particular region of the spectrum. A *graph atom* is a graph signal

that is localized at a certain graph node, and that has certain frequency characteristics imposed by its *kernel* in the spectral domain. A *graph dictionary* is a collection of graph atoms. We adopt a sparse representation of the graph signals over these dictionaries. We propose to learn the spectral structure of the graph dictionaries jointly with the signal representations. The formulation of the graph dictionary problem in the spectral domain permits us to make use of observations from multiple graphs when learning the dictionaries, so that the graph dictionaries can be fit succesfully to a relatively larger set of signals observed over graphs with similar characteristics.

The estimation of the unavailable entries of a graph signals from the available ones is in fact a well-studied problem in machine learning. Many methods in the graph-based semi-supervised learning (SSL) [4] literature address this problem, where the graph signals are assumed to change smoothly on the graph [5], [6], [7], [8], [9], [10]. However, this feature of graph SSL methods inherently impose a strict assumption on the signal spectrum, i.e., the signal is assumed to contain only low frequencies. In this thesis, we aim to obtain a more precise characterization of the spectrum of graph signals by representing them with spectrally concentrated graph kernels. Another difference between our method and this line of works is that graph-based SSL methods would typically need to treat and estimate different graph signals independently of each other, while our algorithm has the flexibility to use data from a single graph or multiple graphs by learning a set of common graph dictionary parameters in the spectral domain, which is enough for generating a graph dictionary for each one of the graphs at hand.

The problem of dictionary learning on graphs has been considered in some previous works before. In [11] a method is proposed for learning graph dictionaries represented with polynomial parameters on multi-graphs. However, the multi-graph signals that are used for learning a dictionary model are assumed to be completely known on the whole graph in [11]. Similarly, the graph signals are assumed to be completely known on the whole graph in works [12], [13], [14], [15]. Compared to these previous works, our study has the important difference that it addresses a more challenging problem where the available graph signals are partially observed, i.e., the signal entries are unavailable on a subset of the graph nodes. Since we are partially observing the signals, we prefer to learn a dictionary model which is expressed with a small number

of parameters. Due to this reason, we prefer to represent the spectrum of the dictionaries using Gaussian kernels, which can be learnt with very few parameters, namely, the mean and the scale of the kernels. Since we can easily control the mean and the scale parameters of the Gaussian kernels, the generated dictionaries can concentrate on the desired frequency range of the spectrum. Several graph atoms generated from different Gaussian kernels are illustrated in Fig. 1.3.

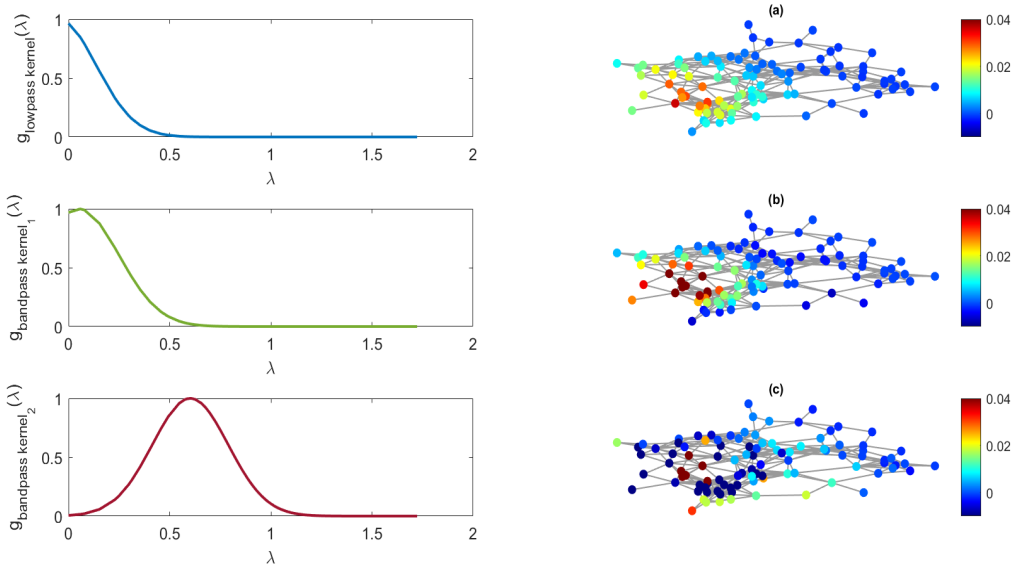


Figure 1.3: At the left side of the figure, a low-pass Gaussian kernel and two different band-pass Gaussian kernels are illustrated. Respectively, (a) is a low-pass dictionary atom and (b), (c) are band-pass dictionary atoms generated from the kernels shown next to them. One can observe that as the spectrum of the atom shifts to high frequencies, its variation over the graph becomes more rapid.

We propose a flexible problem formulation that allows us to learn graph dictionaries in a single graph setting, as well as a multiple graph setting. In the multi-graph setting, the dictionary on each graph consists of a set of spectrally concentrated, graph atoms localized at different graph nodes. The overall dictionary consists of sub-dictionaries, each of which is generated with a Gaussian kernel that captures a different part of the spectrum. The Gaussian kernel parameters defining the dictonaries are common between all graphs, which inherently relies on the assumption that the signal sets

observed on the different graphs at hand bear similar characteristics in terms of their frequency content.

We formulate our problem as a non-convex optimization problem, where the sparse coefficients of the multi-graph signals and the parameters of the Gaussian kernels are learnt jointly. We minimize our objective function with an alternating optimization procedure, where the sparse coding and the dictionary update steps are applied iteratively. In the first step, we fix the graph dictionaries by fixing the Gaussian kernel parameters and solve for the sparse coefficients. In the second step, we fix the sparse coefficients and update the dictionaries by recomputing the Gaussian kernel parameters. At the end of this iterative procedure, finally, the incomplete entries of the given multi-graph signals are estimated using the learnt dictionaries and the sparse coefficients.

1.2 The Outline of the Thesis

The organization of the thesis is as follows: In Chapter 2, we give an overview of the related work. In particular, we discuss the concepts of graph signal processing, semi supervised learning using graph models and dictionary learning. The topic of dictionary learning on graphs is examined in detail.

In Chapter 3, we present our proposed method for the estimation of partially observed ~~multiple~~ graph signals by learning spectrally concentrated graph kernels. We first introduce the setting and the notation, and then formulate the problem. We then describe the sparse representation of graph signals and the method of updating the spectrally concentrated Gaussian graph kernel parameters of graph dictionaries. Finally, the incomplete entries of the input graph signals are estimated based on their representations over the learnt graph dictionaries.

In Chapter 4, experimental results that evaluate the performance of the proposed method on ~~multi-~~graph data sets are presented.

The thesis is concluded with the final discussions, comments, and future work in Chapter 5.

CHAPTER 2

RELATED WORK

This chapter gives a brief overview of graph-based semi-supervised learning algorithms in Section 2.1, we discuss signal processing on graphs in Section 2.2, we give a detailed overview of dictionary learning with dictionary learning in regular domains and dictionary learning on graph domains in Section 2.3.

2.1 Graph Based Semi Supervised Learning Algorithms

In machine learning, the process of learning can be examined under two major titles: supervised and unsupervised [16]. In supervised learning, the goal is to learn a mapping function from labeled input samples x and corresponding output samples y, then construct a classifier which estimates the labels of test samples with the learnt function. In unsupervised learning, the goal is to model the underlying structure or distribution of the data when the labels are unavailable. Semi-supervised learning (SSL) is an area of machine learning that aims to combine these two tasks [17]. Semi-supervised learning uses both labeled and unlabeled data during training. In fact in a graph setting, the estimation of the unavailable entries of a graph signals from the available ones is a well-studied problem in the semi-supervised learning literature. In graph SSL methods, a very typical prior is the smoothness of the graph signal, which implicitly assumes that the graph signal contains rather low frequencies.

Many graph based semi-supervised methods can be considered to estimate a function f on a graph [17]. Two main priors are widely used in graph-based SSL methods: Firstly, the function f should be close to the given labels on the labeled nodes, which is represented as a data fidelity loss term in the objective function. Second, the func-

tion f should vary smoothly on the graph, which is captured through a regularization term in the objective.

The algorithm in [5] uses pairwise relationships of the given labeled data to learn from both labeled and unlabeled data based on graph mincuts. The algorithm does a binary classification that assigns a positive or negative label to given labeled data. A similarity measure is used for constructing the graph. Then unlabeled nodes are classified. The objective function in (2.1) is minimized, which consists of a data fidelity loss function and a regularizer. In the objective function, y_i are the labels to be estimated, L is the given set of labels, $y_{i|L}$ are the available labels, $\mathcal{W}_{ij}$ is the weight matrix that denotes the similarity between nodes i and j. The data fidelity loss is multiplied by a very large valued coefficient μ to fix the labeled data at their given labels.

$$\arg \min_{y_i \in \{0,1\}, \forall i} \mu \sum_{i \in L} (y_i - y_{i|L})^2 + \frac{1}{2} \sum_{i,j} \mathcal{W}_{ij} (y_i - y_j)^2 \tag{2.1}$$

The mincut approach [5] is extended by adding randomness to the graph structure in [6]. In [7], a method is proposed for computing the marginal probabilities of discrete Markov random fields to classify the data in a semi supervised manner. The problem is formulated with a Markov random field model, where nodes are data points and edge weights are set with a local distance metric. The proposed method uses Boltzmann machines to learn parameters which maximize the likelihood of labeled data, then classifies the unlabeled data with the computed labels. Several Markov Chain Monte Carlo [18] sampling methods are suggested for learning such as global Metropolis sampling and Swendsen-Wang sampling.

A well-known method is proposed in [9] for semi-supervised learning with local and global consistency. Here, the local consistency assumption that nearby points are likely to have the same label and the global consistency assumption that points on the same structure are likely to have the same label are important. The method uses a loss function and the Laplacian as a regularizer in its objective in (2.2). In the objective function, y_i is the i^{th} entry of the label y signal to be estimated, $y_{i|L}$ denote the available labels, $\mathcal{W}_{ij}$ is the weight matrix representing the similarity between nodes i and j and $\mathcal{D}$ is the degree matrix. The first term in the objective function can

be regarded as a graph Laplacian based regularization term.

$$\min_{y_i} \frac{1}{2}\Big(\sum_{i,j=1}^{n} \mathcal{W}_{i,j} \left\| \frac{y_i}{\sqrt{\mathcal{D}_{ii}}} - \frac{y_j}{\sqrt{\mathcal{D}_{jj}}} \right\|^2 + \mu \sum_{i=1}^{n} \left\| y_i - y_{i|L} \right\|^2 \Big) \tag{2.2}$$

In [10], the extensions of RLS (Regularized Least Squares) and SVM (Support Vector Machines) methods are proposed based on a manifold model. A data-dependent regularization approach is considered in this work based on the graph Laplacian, which leads to the proposed LapRLS and LapSVM methods.

Semi-supervised learning methods can be inductive or transductive: Inductive SSL uses both the labeled and the unlabeled data and constructs a classifier in the training phase. Then with the help of the classifier, it classifies the test data in the testing phase. Meanwhile, graph-based semi-supervised learning methods are mostly transductive, which consider the whole graph with all labeled and unlabeled data entries and predict the unlabeled data as a set as in [4], [8]. In [8], harmonic functions are used for the semi-supervised classification purpose. The harmonic property assumption is that the function value at each unlabeled data node is the average of the values at its neighboring nodes. The optimization function consists of a convex loss function which fixes the labeled data to the given label values, and a graph Laplacian regularizer term that enforces the learnt function to be smooth on the graph. Similarly, the study in [4] aims to achieve two goals: First, the predicted labels (which are initially unlabeled) should match the true labels. The second goal is to estimate the label function as a smooth function, so that similar data points have similar label predictions. The overall objective function of the method is in (2.3), where $\mathcal{L}$ is the graph Laplacian.

$$\min_{y_i} \sum_{i \in L} (y_i - y_{i|L})^2 + \mu y^T \mathcal{L} y \tag{2.3}$$

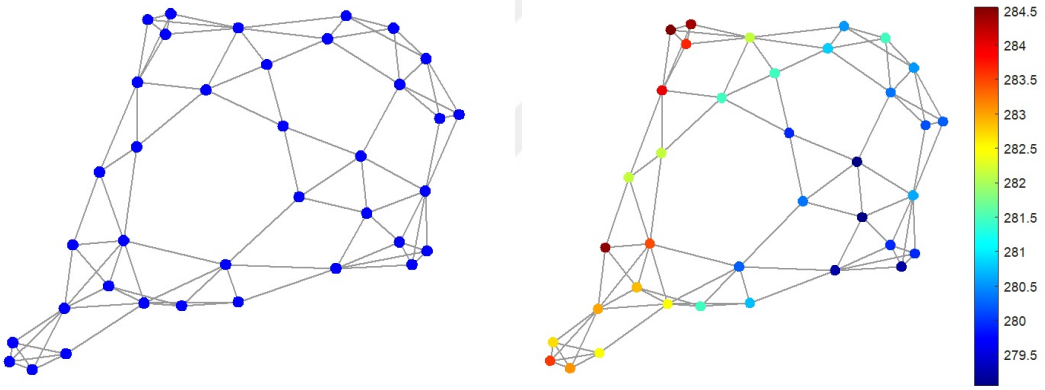
When the output space is discrete, class labels can be predicted with classification methods. When the output space is continuous, regression methods are used for predicting the output values. In graph-based SSL methods, classification on discrete case can be easily extended to regression methods such as [19].

Many of the classification and regression algorithms in the graph-based SSL, assume the graph signals to be estimated change smoothly on the graph, which do not model the band-pass or high-pass variations on their spectra when the signals exhibit some

isolated and local behaviour in certain graph regions. In this thesis, we aim to go beyond the smoothness assumption, which means that graph signal contains mostly low frequecies and obtain better performance than such methods by trying to capture the different spectral properties of the signals in more detail.

2.2 Signal Processing on Graphs

Graphs are often used for modeling data sets with an underlying topological structure, such as energy, transportation, social and sensor networks. Such networks are represented with graph models, where measurements taken over the nodes are regarded as graph signals. In Figure 2.1, a graph structure representing a sensor network and a graph signal that represents temperature measurements taken on this graph are illustrated.



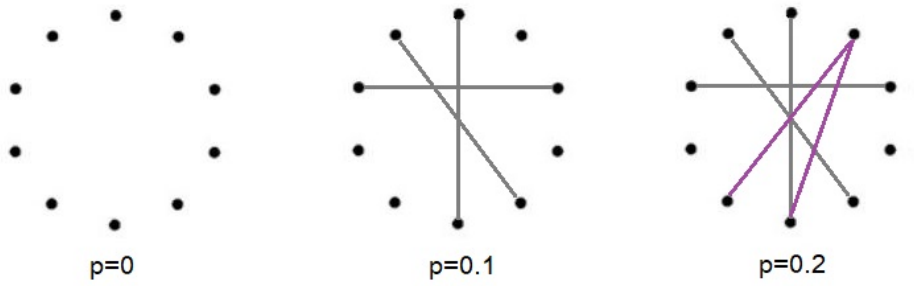
(a) (b)

Figure 2.1: (a) Graph structure for a temperature sensor network, (b) a graph signal whose entries are temperature measurements in *Kelvin* taken at a certain date and hour.

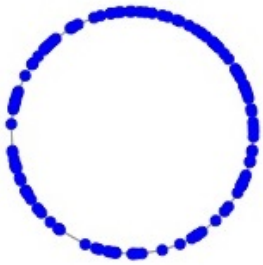
Basically, graphs consist of nodes (also called "vertices") and edges. When we represent a network as a graph, nodes represent entities such as sensors or users where the edges model the interactions between the nodes. If two nodes have similar characteristics, there will be an edge between them, for instance users who graduate from

a school in the same year or like common subjects in a social network, nearby locations with similar temperature or wind speed characteristics in a weather network [2], points on a highway that are affected by the same bottlenecks in a road network, etc. are typically linked with an edge.

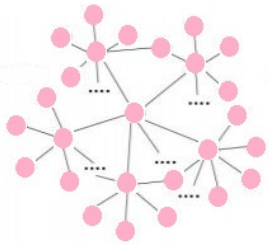
In order to represent real world data with graphs; Erdős–Rényi graphs, ring graphs, random geometric graphs, small-world graphs, power-law graphs, K-nearest-neighbor (KNN) graphs and scale-free graphs [2] are widely used. In Figure 2.2, some graph construction methods are illustrated. When constructing the graphs, edges are usually determined based on the physical proximity of the nodes, from which sensor measurements are taken. However, sometimes there is no information about places, then this knowledge can be inferred from the data using weight matrix.



(a) Erdős–Rényi graphs



(b) Ring graph



(c) KNN graph

Figure 2.2: Illustration of some graph construction methods

The main challenge in graph signal processing is that graphs have complex and irregular structures, and the main consideration is to analyze graph signals in a manner that is aware of the underlying structure. In order to analyze graph signals, the pa-

pers [1], [2] and [3] study how the classical concepts of signal processing such as Fourier transform, filtering, convolution and translation can be extended to graph domains. Following these results, graph signals can be represented in two domains: the vertex domain and the graph spectral domain, i.e., the frequency domain.

The edges between nodes can be directed or undirected in general. In this thesis, we limit our interest to graph structures with undirected edges. Working with directed graphs poses additional challenges, since several properties of the graph Laplacian are different; e.g., the eigenvectors of the graph Laplacian of a directed graph are generally not orthonormal [20].

2.2.1 Weighted Graph and Graph Signals

We represent an undirected, connected, weighted graph as $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, where $\mathcal{V}$ is a finite set of N vertices (nodes), $\mathcal{E}$ is a set of edges, and $\mathcal{W}$ is a weighted adjacency matrix. If there is an edge between nodes i and j, the $(i,j)^{th}$ entry of $\mathcal{W}_{ij}$ the weight matrix represents the weight of the edge and $\mathcal{W}_{ij} = \mathcal{W}_{ji}$ in undirected graphs. For instance, in a sensor network, if two nodes are far from each other, a small weight is assigned to the edge between these nodes, however if these two nodes are close to each other, a higher weight is assigned to the related edge. If there is no edge between the nodes i and j, we set $\mathcal{W}_{ij} = 0$ [3]. We consider a general setting where the graphs edges do not necessarily model the physical proximity between the graph nodes, but may model other kind of abstract relations as well. For instance, in a social network where different nodes represent different users, an edge weight can be assigned as 1 if the corresponding users are friend, otherwise it can be assigned as 0. The edge weights are assumed to be nonnegative in this work.

The edge weights are sometimes defined naturally, however sometimes they are not defined, in which case a common way to define the weight of an edge connecting node i and node j is to use a thresholded Gaussian kernel weighting function:

$$\mathcal{W}_{ij} = \begin{cases} \exp\left(-\frac{|dist_{ij}|^2}{2\theta^2}\right) & dist_{ij} \leq \kappa \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

Here θ and κ are some parameters for determining the weights. $dist_{ij}$ can be a phys-

ical distance or Euclidian distance between node i and node j. Another way to construct a graph is to connect each node to its K-nearest neighbors with respect to the physical or feature space distances. More details on graph construction strategies can be found in [21]. A graph signal y is formally defined as a function $y(v)$ on nodes $v \in \mathcal{V}$ of the graph $\mathcal{G}$.

A graph signal y can be represented as a vector $y \in \mathbb{R}^N$, whose i^{th} entry gives the value of the signal at node i. An example of a graph signal y is shown in Figure 2.3.

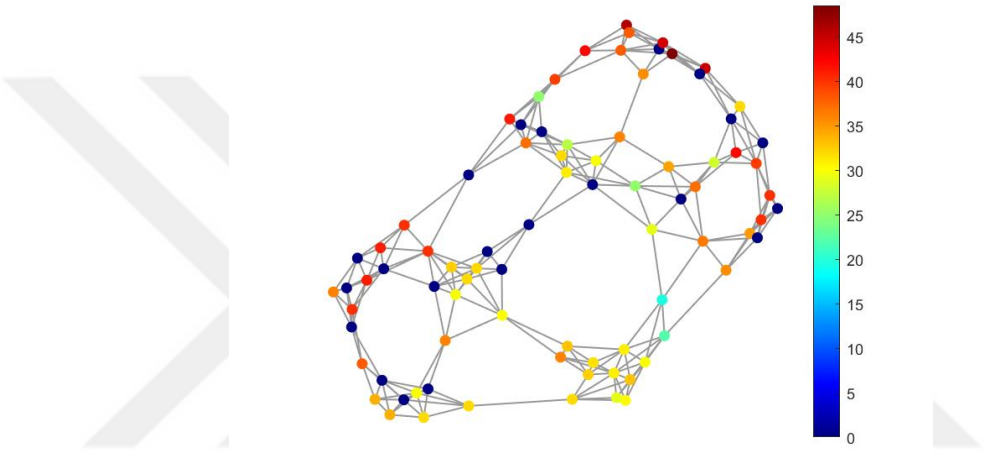


Figure 2.3: A graph signal from a temperature dataset

2.2.2 Graph Laplacian

For a weighted and undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{W})$, the degree matrix $\mathcal{D}$ is defined as the diagonal matrix whose i^{th} diagonal element is equal to the sum of the weights of all the edges connected to node i. Nodes, edges and degrees in a graph network are illustrated in Figure 2.4.

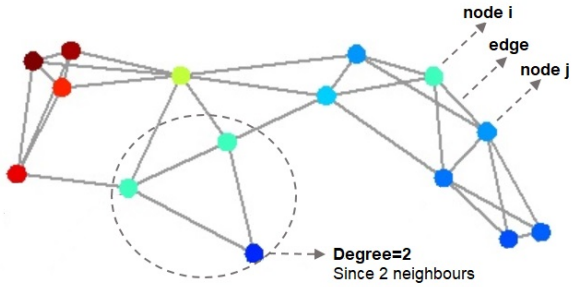


Figure 2.4: Illustration of nodes, edges and degrees

The graph Laplacian is often regarded as the graph equivalent of the Laplace operator in classical signal processing [22]. The graph Laplacian operator is defined as

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

There is also a normalized version of the graph Laplacian, which is defined as

$$\mathcal{L} = \mathcal{I} - \mathcal{D}^{-1/2}\mathcal{W}\mathcal{D}^{-1/2}. \tag{2.6}$$

The normalized graph Laplacian $\mathcal{L}$ is a real symmetric matrix; therefore, it is diagonalizable and it has a complete set of orthonormal eigenvectors. It can be verified from equation (2.6) that the graph Laplacian is a positive semi-definite matrix, which implies that it has nonnegative eigenvalues. The eigenvectors are denoted by $U = [U_0, U_1, ..., U_{N-1}]$, and the eigenvalues $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \leq ... \leq \lambda_{(N-1)} \leq 2$ are often represented with the diagonal matrix

$$\Lambda := \begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix}.$$

Hence, the graph Laplacian has the eigenvalue decomposition:

$$\mathcal{L} = U \begin{bmatrix} \lambda_0 & & 0 \\ & \ddots & \\ 0 & & \lambda_{N-1} \end{bmatrix} U^T \tag{2.7}$$

The eigenvectors of the Laplacian operator are important for the analysis of the graph

signals residing on the graph, and the corresponding eigenvalues carry the notion of frequency [3].

2.2.3 Graph Fourier Transform

As mentioned in Section 2.2.2, the eigenvectors and eigenvalues of the graph Laplacian operator contain useful information for analyzing graph signals. The graph Laplacian is equivalent to the Laplacian operator in classical signal processing. The eigenfunctions of the Laplacian operator are used as Fourier basis. With this idea, the eigenvectors of the normalized graph Laplacian are used as graph Fourier basis. Thanks to this, the graph Fourier transform can be defined for any function $y \in \mathbb{R}^N$ on the vertices of the graph $\mathcal{G}$. The Graph Fourier Transform (GFT) and its inverse provide a way of representing a graph signal both in the vertex domain and the spectral domain. The expansion of y in terms of the eigenvectors of the graph Laplacian is given by

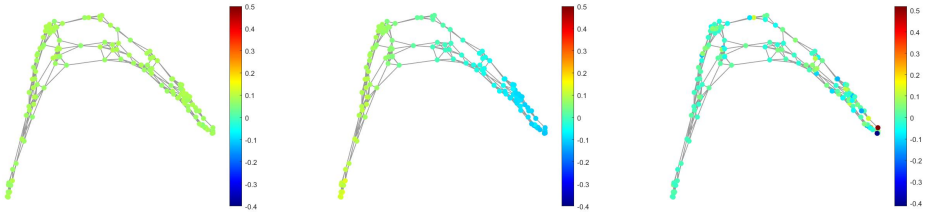
$$\hat{y}(\lambda_l) := \langle y, u_l \rangle = \sum_{i=1}^{N} y(i) u_l^*(i) \tag{2.8}$$

In equation (2.8), y is the graph signal in the vertex domain and $\hat{y}$ represents the Fourier transform of y in the spectral domain. The eigenvectors of the graph Laplacian are denoted by $\{u_l\}_{l=0,1,\ldots,N-1}$. Then, the inverse Fourier transform can be computed as follows:

$$y(i) = \sum_{l=0}^{N-1} \hat{y}(\lambda_l) u_l(i) \tag{2.9}$$

In classical Fourier analysis, the eigenvalues carry a specific notion of frequency: for low frequencies which are close to zero, the relevant complex exponential eigenfunctions are smooth, slowly oscillating functions, whereas for high frequencies far from zero, the relevant complex exponential eigenfunctions oscillate more rapidly. In the graph setting [3], the graph Laplacian eigenvalues and eigenvectors provide a similar notion of frequency. For connected graphs, the Laplacian eigenvector u_0 associated with the 0^{th} eigenvalue is constant at each vertex. The graph Laplacian eigenvectors associated with low frequencies λ_l vary slowly across the graph; for example, if two vertices are connected with an edge having a large weight, the values of the eigenvector at those vertices are likely to be similar. The eigenvectors associated with larger

eigenvalues oscillate more rapidly and their values on neighboring vertices are likely to differ more. This is shown in Figure 2.5, where three different graph Laplacian eigenvectors from a random sensor network graph are shown as graph signals.



(a) u_0 (b) u_1 (c) u_{100}

Figure 2.5: Three eigenvectors of the graph Laplacian of a random sensor network graph. The eigenvector u_0 is constant at each vertex, the eigenvector u_1 varies slowly across the graph whereas u_{100} contains many zero crossings.

2.2.4 Filtering on Graphs

Filtering is a generalized operation on graphs. The frequency content of a graph signal can be modified by amplifying or attenuating its Fourier coefficients through the use of graph filters [23]. Graph filters are used in applications including the analysis [24], [25], classification [26], [27], reconstruction [28], [29], [30] denoising [31], [32], [33] and clustering [34] of graph signals.

Graph filtering is an operation on the graph signal with graph spectral domain output

$$\hat{y}_{output}(\lambda_l) = \hat{y}_{input}(\lambda_l)\hat{h}(\lambda_l) \tag{2.10}$$

where each λ_l frequency in the input is amplified or attenuated multiplied by $\hat{h}(\lambda_l)$.

By applying the inverse GFT on both sides of (2.10), the filtering operation can be expressed in the vertex domain as

$$y_{output}(i) = \sum_{l=0}^{N-1} \hat{y}_{input}(\lambda_l)\hat{h}(\lambda_l)u_l(i). \tag{2.11}$$

Each $\hat{h}(\lambda_l)$ kernel gives us a graph filter. This graph filter can be written as a linear

operator in the form of $\hat{H}(\mathcal{L}) = U\hat{H}(\Lambda)U^T$, where $\hat{H}(\mathcal{L})$ contains all graph filters generated by all eigenvalues λ_l then (2.11) can be written as

$$y_{output} = U\hat{H}(\Lambda)U^T y_{input}. \tag{2.12}$$

2.2.5 Convolution on Graphs

The convolution property in classical signal processing can not be applied directly to the graphs because it is not straightforward to find the equivalent of the time shifting term $h(t-\tau)$ for graph signals, since graphs have an irregular structure. However, one can use the property that multiplication in spectral domain, which is given in (2.10), is equivalent to the convolution in the vertex domain. From equation (2.11) we can write convolution in the vertex domain as

$$(y * h)(i) = \sum_{l=0}^{N-1} \hat{y}_{input}(\lambda_l)\hat{h}(\lambda_l)u_l(i) \tag{2.13}$$

2.2.6 Translation on Graphs

The translation operator on graphs can be defined through convolution with the delta function in the vertex domain as in (2.14), just like in classical signal processing.

$$T_n y = \sqrt{N}(y * \delta_n)(i) = \sqrt{N}\sum_{l=0}^{N-1} \hat{y}_{input}(\lambda_l)\hat{u}_l^*(n)u_l(i) \tag{2.14}$$

where $\delta_n(i) = \begin{cases} 1 & if\ i = n \\ 0 & otherwise. \end{cases}$

2.3 Dictionary Learning

In real world, there are huge amounts of data captured by natural sensors and artificial sensors. These data carry a lot of information, but mostly contain multiple correlated versions of the same information. The relevant part generally has a much smaller dimension when compared to the recorded big data [35] and this reduced version of

the data is quite informative of the underlying process. Dictionary learning algorithms address this problem and involve two main procedures: the sparse coding and the dictionary update steps.

The goal of sparse representation is to express a signal y as a linear combination of a small number of signals (atoms), taken from a resource database that is called a "dictionary". The number of this "small number of signals" is determined by a sparsity constraint. The objective is to find a representation of the signal that contains a small number of nonzero coefficients so as to satisfy the sparsity constraint. This problem is formulated as an optimization problem in (2.15), where Y represents a signal matrix whose i^{th} column y_i is the i^{th} signal to be decomposed, D represents the dictionary, X represents the sparse coefficient matrix of the signal set Y, whose columns are represented by x_i. The problem contains an $\|.\|_0$ norm term, which counts the total number of nonzero elements of a vector. This corresponds to the sparsity constraint.

$$\min_X \sum_{i=1} \|x_i\|_0 \ \ subject\ to\ Y = DX \tag{2.15}$$

As seen in (2.15), the sparse coding problem contains an $\|.\|_0$ norm term, which makes it NP-hard. The algorithms which are used for finding sub-optimal solutions to this problem can be examined in two groups [35]. The first group includes greedy algorithms such as matching pursuit [36], and orthogonal matching pursuit [37] which iteratively choose locally optimal atoms and add them to the representation of the signal at hand. The second group consists of algorithms based on convex relaxation methods such as basis pursuit denoising [38], and least absolute shrinkage and selection operator (LASSO) [39]. Convex relaxation methods give permission to replace the non-convex $\|.\|_0$ norm term by the convex $\|.\|_1$ norm, which results in the new formulation given in (2.16).

$$\min_X \|Y - DX\|_2^2 + \lambda \sum_{i=1} \|x_i\|_1 \tag{2.16}$$

In addition to pursuit algorithms, there exist other sparse representation algorithms such as the focal underdetermined system solver (FOCUSS) [40], and sparse Bayesian Learning [41]. The sparsity of the coefficient vector depends on both the represented signal and the learnt overcomplete dictionary.

This section continues with brief explanations about dictionary learning for signal approximation and classification in regular domains,and also domain adaptative dictionary learning algorithms in regular domains. Finally, we give an overview of dictionary learning on graph domains which is an essential motivation for the proposed algorithm.

2.3.1 Dictionary Learning for Signal Approximation and Classification in Regular Domains

Various approaches exist in the literature for dictionary learning methods for signal approximation in regular domains, such as probabilistic learning methods based on maximum likelihood (ML) [42], the method of optimal directions (MOD) [43] and clustering based methods such as the K-SVD algorithm [44], which learn dictionaries fit to the common specific structures in the data set.

The K-SVD algorithm [44] learns an over-complete dictionary from a set of training signals so as to provide accurate sparse approximations of the signals in the training set. The name of the method is inspired from the K-means clustering algorithm. K-SVD is an iterative method that alternates between the sparse coding stage and the dictionary update stage. The K-SVD method can be coupled with any pursuit algorithm for its sparse coding stage. The algorithm focuses on the representational power of the learnt dictionary without considering the discrimination capability of the dictionary. The dictionary learning problem is formulated as in (2.17), which aims to find the best dictionary which is represented with D, to approximate the training data set Y with sparse representations X. The columns of the sparse code matrix by x_i, and the sparsity constraint for all vectors x_i as T_0 the dictionary learning problem can be formulated as

$$\min_{D,X} \|Y - DX\|_F^2 \; subject\ to\ \forall i, \|x_i\|_0 < T_0. \tag{2.17}$$

Besides the representational power, dictionaries can also be designed and learnt in view of particular applications, such as classification. In [45], a discriminative dictionary and linear classifier are learnt iteratively. The method contains a two-stage

process: First, in the representation stage, the sparse representation X of an input image Y is found based on the dictionary D. Next, the second stage is the classification stage, where a linear classifier V is found with the help of the sparse representation. The constrained optimization problem in (2.18) involves the representational error for both labeled and unlabeled data and a loss term for classification. In the objective function, H_l is the multivariate labels of the labeled training data, and the sparse codes are denoted by X_l for the labeled data and X_u for the unlabeled data. The labeled training images are represented with Y_l, and the unlabeled ones are represented with Y_u. The ρ_l and ρ_u parameters control the trade-off between the representation and the classification accuracies.

$$\arg\min_{D,X} \left\|H_l - V^T X_l\right\|_F^2 + \gamma \left\|V\right\|_F^2 + \rho_l \left\|Y_l - DX_l\right\|_F^2 + \rho_u \left\|Y_u - DX_u\right\|_F^2 \\ subject\ to\ \left\|x_i\right\|_0 \leq T_0 \tag{2.18}$$

At first sight the formulation in [46] can be considered as a special case of [45] that uses only labeled data. The objective function in (2.19) looks like the original K-SVD method. The objective of the proposed method which is called D-KSVD, includes a discriminative term in addition to the reconstruction term of the original K-SVD method. In the objective function, Y is the set of input images, D is the dictionary, X is the sparse coefficient matrix, H contains the labels of the training input images, V is the classifier and the γ parameter controls the contribution of the corresponding term.

$$\arg\min_{D,W,X} \left\| \begin{pmatrix} Y \\ \sqrt{\gamma} * H \end{pmatrix} - \begin{pmatrix} D \\ \sqrt{\gamma} * V \end{pmatrix} * X \right\|_2^2 subject\ to\ \left\|x_i\right\|_0 \leq T_0 \tag{2.19}$$

The dictionary learning algorithm in [47] has both discriminative and reconstructive power. It learns a discriminative dictionary and a linear classifier simultaneously. The formulation in (2.20) [47] is similar to the one in (2.19) [46], but contains an additional term, which is called the label consistency regularization term. This regularization term represents the sparse coding error that enforces classical sparse codes

to approximate the discriminative sparse codes. In the objective function, Y is the set of input signals, D is the dictionary, X is the sparse coefficient matrix, H contains the labels of the training input images, V is the classifier, Q is the discriminative sparse codes of the input signals Y, A is the linear transformation matrix which transforms the original sparse codes X to the discriminative sparse codes Q, and α and β control the relative contributions of the corresponding terms.

$$\arg \min_{D,V,A,X} \| \begin{pmatrix} Y \\ \sqrt{\alpha} * Q \\ \sqrt{\beta} * H \end{pmatrix} - \begin{pmatrix} D \\ \sqrt{\alpha} * A \\ \sqrt{\beta} * V \end{pmatrix} * X \|_2^2 \; subject \; to \; \|x_i\|_0 \leq T_0 \qquad (2.20)$$

Another dictionary learning method is proposed in [48] for pixelwise image classification, which learns dictionaries that are both reconstructive and discriminative. Meanwhile, multiple dictionaries are learnt for the pixelwise classification of image patches.

In contrast to classical dictionary learning algorithms, in [49] the dictionary is directly constructed from training signals for face recognition. The proposed SRC (Sparse Representation-based Classification) algorithm is based on the principle that, if sufficiently many training signals are available from each class, the test signals can be represented as a linear combination of the training signals from the same class. Test signals are sparsely represented using a small number of training signals that are chosen from the whole training database.

2.3.2 Domain Adaptive Dictionary Learning in Regular Domains

In our work, we study the problem of dictionary learning in multiple graph domains. Since we analyze signals in more than one domain, our work has some links to the subject of dictionary learning on multiple domains. Hence, we give a brief overview of domain adaptive dictionary learning in this section. Within the context of dictionary learning, domain adaptive dictionary learning is an important topic that studies how knowledge can be transferred between different domains when the signals at hand are observed from more than one domain. Domain adaptation refers to the

machine learning problem where training samples come from one or more "source domains" while test samples come from a different but related "target domain".

A binary classification method across different domains is proposed in [50]. The labeled data is available in a source domain for training, however the unlabeled test data comes from a different target domain. The proposed method aims to classify unlabeled data in the target domain with the help of the knowledge transferred from the source domain. The objective function is written based on a normalized cut cost function [51], where the knowledge is transferred through a constraint matrix whose columns are n-dimensional vectors with two nonzero entries. Each column has an entry of $+1$ in the i^{th} row, an entry -1 in the j^{th} row, and the other entries are zero. This approach enforces the i^{th} and the j^{th} data samples to have the same label. The method is tested on textual data from different domains.

Domain adaptation can be applied in a semi supervised manner when the source domain contains both labeled and unlabeled data. In [52], a semi-supervised kernel matching method for domain adaptation is proposed. In the source domain, the algorithm learns a prediction function based on a Hilbert Schmidt Independence Criterion (HSIC) [53]. With the help of a mapping matrix, the target domain points are mapped to similar source domain points by matching their geometrical similarities which are represented with kernel matrices. In the optimization problem, the smoothness is enforced through graph Laplacian regularization terms in both the source and the target domains.

In the recent years, various dictionary learning methods have been proposed for the domain adaptation problem. In [54], an approach is proposed for learning a pair of dictionaries for both the source and the target domains, as well as a transformation matrix M that measures the divergence across the two domains. The formulation of objective function is similar to the equation (2.20). The label consistent K-SVD (LC-KSVD) in [54] minimizes a label consistency term, the classification error term of a linear predictive classifier and cross domain loss term, which are given in (2.21). Eventually, this weakly supervised dictionary learning method learns a reconstructive, discriminative and domain adaptive dictionary pair D_t and D_s and also the corresponding classifier parameters. In equation (2.21), Y_t and Y_s represent the target

domain and source domain instances, respectively. The other parameters are the same as in equation (2.20).

$$\arg \min_{D_t, D_s, X_t, V} \| \begin{pmatrix} Y_t \\ Y_s M^T \\ \sqrt{\alpha} * Q \\ \sqrt{\beta} * H \end{pmatrix} - \begin{pmatrix} D_t \\ D_s \\ \sqrt{\alpha} * A \\ \sqrt{\beta} * V \end{pmatrix} * X_t \|_2^2 \; subject \; to \; \left\| x_t^i \right\|_0 \leq T_0 \quad (2.21)$$

In [55], an unsupervised domain adaptative dictionary learning method is proposed. A linear mapping is constructed between the sparse codes of the source and the target domain samples, which contains the information of the nearest couples of the original data in the two domains. With this transformation matrix, a pair of dictionaries are learnt for the source and the target domains, as well as an identical sparse representation for the two domains. The problem formulation then simplies to a K-SVD problem.

The proposed method in [56] learns a dictionary with a K-step process. The dictionary D_K learnt at step K has representative and discriminative power in both the source and the target domains. The objective does not contain any mapping matrix; instead, it learns sparse codes for the source and the target domains separately and a common domain-invariant dictionary.

2.3.3 Dictionary Learning on Graph Domains

In several recent works, the concept of dictionary learning has been extended to graph domains. Similarly to classical dictonary learning, the purpose in graph dictionary learning algorithms is to learn a collection of graph signal prototypes, called graph atoms, using which a given set of graphs signals can be represented sparsely. The methods we will mention in this part are related to the topic of dictionary learning on graph domains, which have great importance for our study.

A structured overcomplete dictionary obtained from a number of sub-dictionaries (or shift invariant filters) is defined in [12]. Every sub-dictionary is defined based on a

certain nonparametric form in the spectral domain, which captures a different spectral component of the training signals Y. In the dictionary update stage, every sub-dictionary is updated at a time while other sub-dictionaries are fixed. The objective function can be seen in equation (2.22). Sparse coefficients are represented by X.

$$\min_{D,X} \|Y - DX\|_F^2 \; subject\ to \; \|x_j\|_0 \leq T_0, \; \forall j \tag{2.22}$$

Here $\{D_i\}_{i=1}^S$ are S sub-dictionaries whose form is given in (2.23). $\{\Lambda_i\}_{i=1}^S$ are some diagonal, positive semi-definite matrices which include shift invariant filters and $\{U\}_{n=1}^N$ are the eigenvectors of the graph Laplacian.

$$\begin{aligned} D &= [D_1, D_2, \ldots, D_S] \\ &= [U\Lambda_1 U^T \; , \; U\Lambda_1 U^T \; , \; \ldots \; , \; U\Lambda_S U^T] \end{aligned} \tag{2.23}$$

In [15], the dictionary is constructed by sub-dictionaries D_s generated from a graph heat kernel approximated with polynomial coefficients resulting from a K-th order Taylor approximation. The objective function is defined like in [12]; however, a new constraint is added that controls the graph sparsity with the number of nonzero edges on the graph through the weight matrix. Based on the learnt weight matrix $\mathcal{W}$, the Laplacian matrix $\mathcal{L}$ is updated in each iteration as well as the dictionary. The objective function can be seen in (2.24) where β_W controls the graph sparsity, α_{sk} denotes the polynomial coefficients.

$$\begin{aligned} \min_{\mathcal{W}} \|Y - DX\|_F^2 + \beta_W \|\mathcal{W}\|_1 \\ subject\ to\ D = [D_1, D_2, \ldots, D_S], \\ D_s = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}^k, \forall s \in \{1, \ldots, S\} \\ \mathcal{L} = I - \mathcal{D}^{-1/2} \mathcal{W} \mathcal{D}^{-1/2} \\ \mathcal{W}_{ij} = \mathcal{W}_{ji} \geq 0, \; \forall i, j, i \neq j \\ \mathcal{W}_{ii} = 0, \forall i. \end{aligned} \tag{2.24}$$

An approach for parametric dictionary learning is proposed in [13] and [14]. These algorithms aim to model graph signals as a combination of overlapping local patterns

which are called sub-dictionaries. The learnt overall dictionary is the concatenation of the sub-dictionaries which are polynomials of the graph Laplacian. A K-order polynomial kernel $\hat{g}$, which is the function of the normalized Laplacian eigenvalues, is translated to each vertex i in the graph as in equation (2.25). The translation of the polynomial $\hat{g}$ to each vertex i, which is Tg, yields a set of N localized atoms for each sub-dictionary.

$$Tg = \sqrt{N}\hat{g}(\mathcal{L}) = \sqrt{N}U\hat{g}(\Lambda)U^T = \sqrt{N}\sum_{k=0}^{K}\alpha_k\mathcal{L}^k \tag{2.25}$$

Here, the graph kernel $\hat{g}$ is defined on the spectral domain, Λ is the diagonal matrix of the eigenvalues, and U is the matrix containing the eigenvectors of the normalized Laplacian. The learnt dictionary is the concatenation of S sub-dictionaries; where each sub-dictionary D_s has the form

$$D_s = \hat{g}_s(\mathcal{L}) = U\big(\sum_{k=0}^{K}\alpha_{sk}\Lambda^k\big)U^T = \sum_{k=0}^{K}\alpha_{sk}\mathcal{L}^k. \tag{2.26}$$

The algorithm learns the polynomial coefficients α that model the dictionary D, jointly with the sparse representations X of the training signals $Y \in \mathbb{R}^{N\times M}$ by solving the following optimization problem:

$$\begin{aligned}
&\min_{\alpha\in\mathbb{R}^{(K+1)S}, X\in\mathbb{R}^{SN\times M}} \{\|Y - DX\|_F^2 + \mu\,\|\alpha\|_2^2\} \\
&\textit{subject to}\quad \|x_m\|_0 \le T_0,\ \forall m \in \{1,\dots,M\}, \\
&\qquad 0 \le D_s \le c,\ \forall s \in \{1,2,\dots,S\}, \\
&\qquad (c-\epsilon)I \le \sum_{s=1}^{S} D_s \le (c+\epsilon)I
\end{aligned} \tag{2.27}$$

Here in (2.27), two constraints are imposed on the kernels; first constraint requires that the kernels are non-negative and uniformly bounded by a given constant c with the inequality $0 \le D_s \le c$, it means maximum eigenvalue of each subdictionary D_s is upper bounded by c. Second constraint aims that the learnt kernels should cover spectrum because the training signals usually contain frequency components which are spread across the entire spectrum. ϵ is a small positive constant, with the formula $(c-\epsilon)I \le \sum_{s=1}^{S} D_s \le (c+\epsilon)I$, authors aim to adjust the frequency range wide enough to express the training signals.

A dictionary learning algorithm for graph signals on multi-graphs is proposed in [11]. The objective is to learn a common structure for different graph dictionaries that can represent graph training signals sparsely on multi graphs. A dictionary structure that is a polynomial function of the eigenvalues and the eigenvectors of the normalized Laplacian is learnt. The coefficients of the polynomials determine the kernels that generate the subdictionaries by defining their spectral content. Similarly to [13], [14], the study in [11] examines the dictionary learning problem in a multi-graph setting as well. The multi-graph dictionary learning problem is cast as an optimization problem in (2.28), where D_t is the structured graph dictionary for a single graph, $Y_t \in \mathbb{R}^{N \times M_t}$, $t = \{1, 2, \ldots, T\}$, T is the number of graphs and X_t^m corresponds to column m of the coefficient matrix, $m = \{1, 2, \ldots, M_t\}$.

$$\begin{aligned} \min_{\alpha \in \mathbb{R}^{(K+1)S}, X_t \in \mathbb{R}^{SN \times M_t}} & \{\sum_{t=1}^{T} \frac{1}{M_t} \|Y_t - D_t X_t\|_F^2 + \mu \|\alpha\|_2^2\} \\ subject\ to \quad & \|X_t^m\|_0 \leq T_0,\ \forall m \in \{1, \ldots, M_t\}, \\ & D_t^s = \sum_{k=0}^{K} \alpha_{sk} \mathcal{L}_t^k,\ \forall s \in \{1, 2, \ldots, S\}, \\ & 0 \leq D_t^s \leq c,\ \forall s \in \{1, 2, \ldots, S\} \end{aligned} \tag{2.28}$$

Here, the constraint $0 \leq D_s \leq c$ requires that the kernels are non-negative and uniformly bounded by a given constant c.

An unsupervised dictionary learning algorithm is proposed in [57] that considers the underlying structure of the data in both the feature and the manifold domains based on graph smoothness constraints. The method jointly learns the graph Laplacian and the graph dictionary. Two regularization terms are used in the objective function. The purpose of the first regularization term, is to ensure that the dictionary atoms vary smoothly on the graph. This regularization term involving the graph Laplacian $\mathcal{L}$ also inherently implies the smoothness of the signals represented with this dictionary. The purpose of the second regularization term $\mathcal{L}_c$ which is the manifold Laplacian, is to satisfy the smoothness of the sparse codes. The formulation of the objective function is given in (2.29), to which authors refer as the $graph^2DL$ method.

$$\min_{L,D,X} \|Y - DX\|_F^2 + \alpha Tr(D^T \mathcal{L} D) + \beta Tr(X \mathcal{L}_c X^T) + \mu \|\mathcal{L}\|_F^2$$
$$subject\ to\ \|x_i\|_0 \leq T_0\ \forall i$$
$$\mathcal{L}_{ij} = \mathcal{L}_{ji} \leq 0 (i \neq j)$$
$$\mathcal{L}\underline{1} = \underline{0}\ \ Tr(\mathcal{L}) = N \qquad (2.29)$$

Here, N is the number of graph nodes, $\underline{0}$ and $\underline{1}$ denote the constant vectors of all-zeros and all-ones.

A dictionary construction method with spectral graph wavelets (SGWT) is proposed in [58]. From the graph Laplacian, its complete set of orthonormal eigenvectors and its nonnegative eigenvalues, the method generates a "mother" wavelet, and produces a desired number of wavelets by scaling this mother wavelet with a scaling factor, which are the functions of normalized graph Laplacian eigenvalues. Translating all these wavelets to each vertex i of the graph yields a set of atoms which correspond to a dictionary. One important property of this method is that, when SGWT designs a dictionary, it uses the knowledge of only the graph topology (i.e., the graph Laplacian), and it does not adapt the dictionary to the taken graph signal measurements unlike the previously mentioned works.

2.3.4 Dictionary Learning in Regular Domains that Model Data with Graphs

In this section we give a brief explanation on dictionary learning methods in regular domains that model the data with graphs by using the techniques such as graph Laplacian regularization. The dictionaries are learnt for signals in regular domains, e.g., image signals in these methods.

One of the first studies in this topic is [59], where an algorithm called graph regularized sparse coding (GraphSC) is proposed. The algorithm builds a K-nearest neighbor graph to visualize the geometrical structure of the data. Using spectral graph theory techniques, a graph regularization term using the graph Laplacian is employed to maintain the local manifold structure. This term is inserted into the sparse coding objective function, so that the obtained sparse representation varies smoothly on the

graph.

The Transfer Sparse Coding (TSC) Algorithm presented in [60] aims to learn a dictionary and the sparse codes on cross domains which contain labeled and unlabeled data. The main idea in TSC is to compute the sparse codes as in [59], while additionally using a MMD (Maximum Mean Discrepancy) regularization for transferring the knowledge between different distributions. MMD is a nonparametric distance measure that compares different distributions.

A supervised dictionary learning algorithm is proposed in [61], which can be considered as the extended version of the method $graph^2DL$ in [57] by applying the idea in the LC-KSVD approach in [47]. The authors replace the label consistency term with a graph regularization term in the problem formulation of the LC-KSVD method. With this modification, the method proposed in [61] aims to achieve better classification performance thanks to the employed smoothness constraints.

In [62], the dictionary based classification algorithms in [57], [59], [61] are extended to a multi-label classification setting. The authors suggest that the relevant categories can be distinguished from irrelevant ones with an adaptive classification threshold in the multi-label setting. Additionally, the proposed algorithm encourages similar signals to have similar sparse codes by using a graph Laplacian regularization term and learning a common dictionary.

CHAPTER 3

PROPOSED METHOD FOR ESTIMATING PARTIALLY OBSERVED GRAPH SIGNALS

In this study, we propose to model graph signals using graph dictionaries whose atoms concentrate at a particular region of the spectrum. We consider a setting where graph signals are partially observed, which means that graph signals are not available over the whole graph but on a subset of the graph nodes. To handle difficulties arising from the lack of data, we use a multi-graph approach, which allows the usage of data observed on more than one graph when learning a common dictionary model. We jointly learn a common graph dictionary model in the frequency domain and find the sparse coefficients of multi-graph signals according to the dictionaries learnt. Then, the unavailable entries of the graph signals at hand are estimated using the learnt dictionaries and the sparse coefficients. As we mentioned Chapter 2, in some works on dictionary learning on graphs [11], [12], [13], [14], [15] the graph dictionary is obtained as the concatenation of some subdictionaries. This is also valid for our proposed work. The choice of the spectral kernels generating the subdictionaries in the main dictionary is an important issue for our case because we are partially observing the graph signals. At this point, we prefer using Gaussian kernels, which can be learnt with very few parameters that are the mean and the scale. Suitably fitting the mean and the scale parameters of the Gaussian kernel to the graph signal data at hand, we can generate subdictionaries that are concentrated on the desired frequency range of the spectrum.

In the graph-based semi-supervised learning literature the estimation of unavailable entries of a graph signal from the available ones is a well-studied problem [4], [5], [6], [7], [8], [9], [10]. These methods assume that the graph signals change on the

graph smoothly, however in the real world, the graph may contain a small, isolated group of nodes, over which the graph signals may have different characteristics from the rest of the nodes. This typically creates band-pass or high-pass components in the observed graph signals. In our work, we aim to model not only low pass components but also these band-pass and high-pass components which lead to local variations in the spectrum. We model each one of these components with a different subdictionary that concentrates over a certain region of the graph spectrum.

We cast an optimization problem that involves finding the sparse representations of graph signals and learning a dictionary which contains spectrally concentrated Gaussian graph kernels. This problem is not convex, so we minimize the objective function with an iterative, alternating optimization procedure. Firstly, we fix the graph dictionary by fixing the Gaussian kernel parameters and update the sparse coefficients; secondly, we fix the sparse coefficients and update the dictionary by recomputing the Gaussian kernel parameters. Finally, the missing entries of graph signals are estimated from the learnt dictionaries and the sparse coefficients.

3.1 Graph Dictionary Model

Our aim in graph dictionary learning is to learn a set of Gaussian kernel parameters that are common for the multiple graphs to construct a structured dictionary on each graph that includes several Gaussian kernel functions capturing different spectral components of the graph signals.

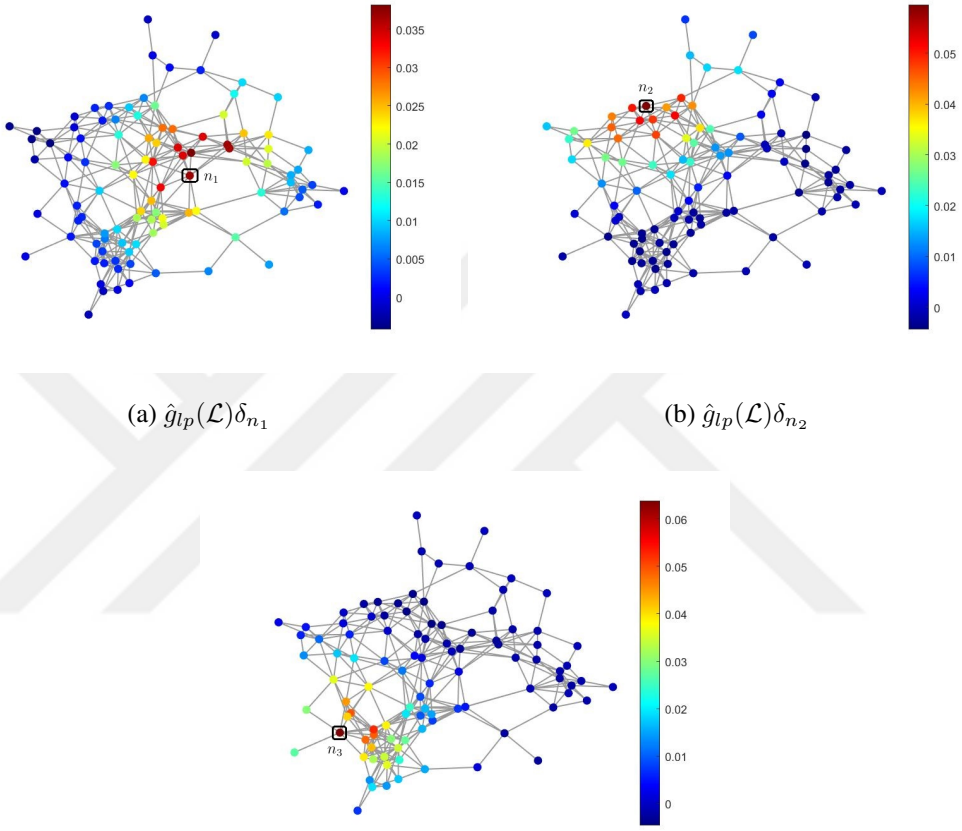
Let $\hat{g}_j(\lambda)$ be a graph kernel where $j = 1, 2, \ldots, J$. Each $\hat{g}_j(\lambda)$ is written in a parametric form with the associated parameter vector $\theta_j = (\mu_j, s_j)$, where μ_j represents the mean parameter and s_j represents the scale parameter of the Gaussian kernels and λ denotes the graph frequency variable.

$$\hat{g}_j(\lambda) = e^{-\left(\frac{\left\|\lambda-\mu_j\right\|^2}{s_j^2}\right)} \tag{3.1}$$

As mentioned in Section 2.2.6, the translation of the kernel $\hat{g}_j(\lambda)$ with the formula (2.13) generates a subdictionary D_j^m when applied to the eigenvalues of the graph Laplacian of graph m as

$D_j^m = U^m \hat{g}_j(\Lambda^m)(U^m)^T \in \mathbb{R}^{N_m \times N_m}$ where $\mathcal{L}^m = U^m \Lambda^m (U^m)^T$.

The different columns (atoms) of a subdictionary represent the signals obtained by localizing the graph kernel at different nodes; e.g., the n^{th} column of the subdictionary represents a graph kernel localized at node n. In Figure 3.1, a low-pass (lp) graph kernel is localized on different graph nodes.



(a) $\hat{g}_{lp}(\mathcal{L})\delta_{n_1}$

(b) $\hat{g}_{lp}(\mathcal{L})\delta_{n_2}$

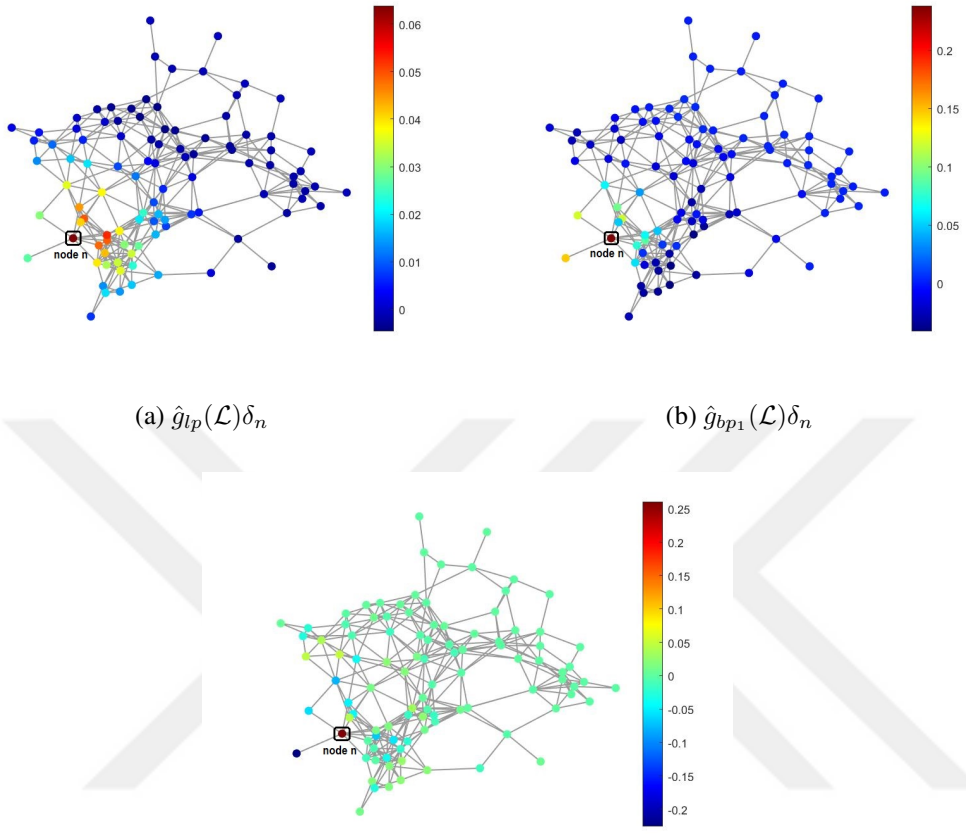
(c) $\hat{g}_{lp}(\mathcal{L})\delta_{n_3}$

Figure 3.1: Three atoms of a subdictionary are illustrated. These atoms are generated from the same low-pass graph kernel by localizing it on different graph nodes. The graph nodes are shown with black boxes.

A structured dictionary is a concatenation of J subdictionaries:

$$D^m = \begin{bmatrix} D_1^m & D_2^m & \cdots & D_J^m \end{bmatrix}$$

In Figure 3.2, three different sub-dictionary atoms are illustrated, which are generated from a low-pass (lp) kernel and two different band-pass (bp) kernels when the kernels are localized on the same node n.



(a) $\hat{g}_{lp}(\mathcal{L})\delta_n$

(b) $\hat{g}_{bp_1}(\mathcal{L})\delta_n$

(c) $\hat{g}_{bp_2}(\mathcal{L})\delta_n$

Figure 3.2: Three different sub-dictionary atoms which are localized on the same node n are illustrated. The sub-dictionaries generated from (a) a low-pass kernel and (b), (c) two different band-pass kernels.

3.2 Graph Signal Model

We begin with describing how we model the graph signal Y^m on graph m, which allows us to better motivate the proposed method. We consider a setting with independently constructed multiple graphs $\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_M$ where each graph $\mathcal{G}_m = (\mathcal{V}_m, \mathcal{E}_m, \mathcal{W}_m)$, for $m = 1, 2, ...M$ is undirected, connected and weighted with edges

$\mathcal{E}_m$ and weight matrix $\mathcal{W}_m$. The number of nodes on graph $\mathcal{G}_m$ is denoted by N_m. We assume that a set of K_m graph signals is observed on each graph $\mathcal{G}_m$.

$$Y^m = \begin{bmatrix} \vdots & \vdots & \cdots & \vdots \\ y_1^m & y_2^m & \cdots & y_{K_m}^m \\ \vdots & \vdots & \cdots & \vdots \end{bmatrix} \in \mathbb{R}^{N_m \times K_m}$$

The graph signal y_i^m on graph m is modeled as

$$y_i^m = \begin{bmatrix} D_1^m & D_2^m & \cdots & D_J^m \end{bmatrix} x_i^m + w_i^m = D^m x_i^m + w_i^m \tag{3.2}$$

where D^m denotes the dictionary, x_i^m represents the sparse coefficients, and w_i^m denotes the noise signal. Each graph dictionary D^m consists of J subdictionaries which are generated from spectrally concentrated Gaussian graph kernels.

Each graph signal $y_i^m \in \mathbb{R}^{N_m}$ on graph m is partially observed, such that only the entries $\{i_l\}$ where $l = 1, 2, ...L_m$ of y_i^m are available.

$$(y_i^m)^T = \begin{bmatrix} ? & ? & y_{i_1} & ? & y_{i_2} & y_{i_3} & ? & ? & \cdots ? & y_{i_{L_m}} \cdots \end{bmatrix} \tag{3.3}$$

The Gaussian kernel parameters with the parameter vector θ_j are sampled from a normal distribution $\mathcal{N}(m_{\underline{\theta}}, \Sigma_{\underline{\theta}})$ with density $f_{\underline{\theta}}(\theta)$, where $m_{\underline{\theta}}$ is a mean vector and $\Sigma_{\underline{\theta}}$ is a diagonal covariance matrix. We assume θ_j's are i.i.d. with density $f_{\underline{\theta}}(\theta)$.

The entries of x_i^m are independently sampled from a Laplace distribution with parameter δ with density $f_{\underline{x}(x)}$. The noise signal w_i^m is sampled from a normal distribution $\mathcal{N}(0, \sigma^2)$ with density $f_{\underline{w}(w)}$.

The joint distribution of Y^m, the Gaussian kernel parameter vector

$\psi^T = \begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_J & s_1 & s_2 & \cdots & s_J \end{bmatrix}$ and X^m is given by $f_{\underline{Y},\underline{\psi},\underline{X}}(Y, \psi, X)$. Assuming the independence of $\psi, X$and w's

$$\begin{aligned} f_{\underline{Y},\underline{\psi},\underline{X}}(Y, \psi, X) &= f_{\underline{\psi}}(\psi) f_{\underline{X}|\underline{\psi}}(X|\psi) f_{\underline{Y}|\underline{X},\underline{\psi}}(Y|X, \psi) \\ &= f_{\underline{\psi}}(\psi) f_{\underline{X}}(X) f_{\underline{Y}|\underline{X},\underline{\psi}}(Y|X, \psi) \end{aligned} \tag{3.4}$$

$$f_{\underline{\psi}}(\psi) = \prod_{j=1}^{J} f_{\underline{\theta}}(\theta_j) = \prod_{j=1}^{J} \frac{1}{\sqrt{(2\pi)^{2J}|\Sigma_{\underline{\theta}}|}} \exp\left(-\frac{1}{2}(\theta_j - m_{\underline{\theta}})^T \Sigma_{\underline{\theta}}^{-1} (\theta_j - m_{\underline{\theta}})\right) \tag{3.5}$$

$$
\begin{aligned}
f_{\underline{X}}(X) &= \prod_{m=1}^{M}\prod_{i=1}^{K_m} f_{\underline{x}}(x_i^m) = \prod_{m=1}^{M}\prod_{i=1}^{K_m}\prod_{l=1}^{JN_m} \frac{1}{2\delta}\exp\left(-\frac{|x_i^m(l)|}{\delta}\right) \\
&= \prod_{m=1}^{M}\prod_{i=1}^{K_m} \frac{1}{(2\delta)^{JN_m}}\exp\left(-\frac{\|x_i^m\|_1}{\delta}\right) = \frac{1}{(2\delta)^{J(\sum_{m=1}^{M} N_m K_m)}}\exp\left(-\frac{\|X\|_1}{\delta}\right)
\end{aligned}
\tag{3.6}
$$

where $X = [X^1\ X^2\ \cdots\ X^M]$ contains the sparse coefficients of all graphs.

$$
\begin{aligned}
f_{\underline{Y}|\underline{X},\underline{\psi}}(Y|X,\psi) &= \prod_{m=1}^{M}\prod_{i=1}^{K_m} f_{\underline{w}}(y_i^m - D^m x_i^m) \\
&= \prod_{m=1}^{M}\prod_{i=1}^{K_m} \frac{1}{\sqrt{(2\pi)^{N_m}\sigma^{2N_m}}}\exp\left(-\frac{1}{2\sigma^2}\|y_i^m - D^m x_i^m\|^2\right) \\
&= \frac{1}{(2\pi\sigma^2)^{\frac{1}{2}\sum_{m=1}^{M} N_m K_m}}\exp\left(-\frac{1}{2\sigma^2}\sum_{m=1}^{M}\sum_{i=1}^{K_m}\|y_i^m - D^m x_i^m\|^2\right)
\end{aligned}
\tag{3.7}
$$

3.3 Proposed Method and Objective Function

In order to get a MAP estimate of the parameters $\underline{X}$ and $\underline{\psi}$, we maximize the log likelihood for the distribution $f_{\underline{\psi}}(\psi) f_{\underline{X}}(X) f_{\underline{Y}|\underline{X},\underline{\psi}}(Y|X,\psi)$ in (3.4) which is equivalent to minimizing the objective in (3.8).

$$
L(X,\psi) = \eta_\theta \sum_{j=1}^{J}(\theta_j - m_{\underline{\theta}})^T \Sigma_{\underline{\theta}}^{-1}(\theta_j - m_{\underline{\theta}}) + \eta_x \sum_{m=1}^{M}\|X^m\|_1 + \eta_w \sum_{m=1}^{M}\|Y^m - D^m X^m\|_2^2
\tag{3.8}
$$

Here, η_θ, η_x and η_w are the trade-off parameters which control the contribution of the corresponding terms.

We assume that due to practical problems such as broken sensors, connection loss, or storage problems, we can not take measurements from some of the sensors, which correspond to a subset of the graph nodes. This situation causes some entries to be missing on the graph signals, which means that we have partially observed the graph signals Y^m. We only know some of the rows of the graph signal matrix Y^m. For selecting these known rows of the graph signal set, we use a binary selection

mask matrix S^m for each graph. The details of the binary selection mask matrices will be given in Section 3.3. Restricting the log-likelihood function in (3.8) only to the available observations with the help of the binary selection matrices, our new objective function becomes

$$L(X,\psi) = \eta_\theta \sum_{j=1}^{J} (\theta_j - m_{\underline{\theta}})^T \Sigma_{\underline{\theta}}^{-1} (\theta_j - m_{\underline{\theta}}) + \eta_x \sum_{m=1}^{M} \|X^m\|_1 + \eta_w \sum_{m=1}^{M} \|S^m Y^m - S^m D^m X^m\|_2^2 \tag{3.9}$$

If we assume that the random variables $\begin{cases} \mu_j \sim \mathcal{N}(0, \sigma_{\mu_j}^2) \\ s_j \sim \mathcal{N}(s_0, \sigma_{s_j}^2) \end{cases}$ are independent, their mean vector and covariance matrix can be written as $m_{\underline{\theta}} = \begin{bmatrix} 0 \\ s_0 \end{bmatrix}$, $\sum_{\underline{\theta}} = \begin{bmatrix} \sigma_{\mu_j}^2 & 0 \\ 0 & \sigma_{s_j}^2 \end{bmatrix}$. Then the objective function in (3.9) can be simplified to the following objective

$$\arg\min_{X,\psi} \eta_\mu \sum_{j=1}^{J} (\mu_j)^2 + \eta_s \sum_{j=1}^{J} (s_j - s_0)^2 + \eta_x \sum_{m=1}^{M} \|X^m\|_1 + \eta_w \sum_{m=1}^{M} \|S^m Y^m - S^m D^m X^m\|_F^2 \tag{3.10}$$

where η_μ and η_s are the trade-off parameters that control the contribution of the first two terms in (3.10). From now on, we take $\eta_\mu = 1$ and the objective function becomes

$$\arg\min_{X,\psi} \sum_{j=1}^{J} (\mu_j)^2 + \eta_s \sum_{j=1}^{J} (s_j - s_0)^2 + \eta_x \sum_{m=1}^{M} \|X^m\|_1 + \eta_w \sum_{m=1}^{M} \|S^m Y^m - S^m D^m X^m\|_F^2 . \tag{3.11}$$

In the objective function, the first term is a regularization term on the mean values μ_j's of the Gaussian kernels, which aims to prevent the spectrum of the Gaussian kernels from shifting to very high frequencies. The second term guides the Gaussian kernel scale parameters s_j's towards a predetermined value s_0, which may be set as a nominal bandwidth value depending on the application at hand. The third term aims to limit the total number of non-zero entries of the coefficient vectors of the graph signals, in order to ensure the sparsity of the representation. The last term ensures that the multiplication of the learnt dictionary and the sparse coefficients is close to the original graph signal set.

As one can see from the objective function in (3.11), the entries of the last term are multiplied with S. These S^m's are binary selection mask matrices that consist of 0's and 1's, and they extract the known entries of the graph signals on graph m. When the graph signal vector in (3.2) is multiplied with the binary selection mask matrix, it becomes

$$(S^m y_i^m)^T = \begin{bmatrix} y_{i_1} & y_{i_2} & y_{i_3} & \cdots & y_{i_{L_m}} \end{bmatrix}. \tag{3.12}$$

The objective function in (3.11) is a convex function of the sparse coefficients, however it is a non-convex, smooth function of the Gaussian kernel function parameters which are used for generating the subdictionaries. Hence, we deal with a non-convex optimization function whose global minimum we cannot find. We hence propose a constructive solution based on a local optimization procedure.

We propose to minimize the objective with an iterative and two-stage alternating optimization approach, by first fixing the Gaussian kernel parameters, which means fixing the dictionaries and updating the coefficients X^m; and then fixing the coefficients X^m and updating the Gaussian kernel parameters in each iteration. We thus minimize the objective function with the following alternating optimization procedure:

Step 1: Fix ψ, optimize X^m

$$\min_{\{X^m\}_{m=1,2,...,M}} \sum_{m=1}^{M} \sum_{i=1}^{K_m} \eta_x \left\| x_i^m \right\|_1 + \sum_{m=1}^{M} \sum_{i=1}^{K_m} \eta_w \left\| S^m y_i^m - S^m D^m x_i^m \right\|_F^2 \tag{3.13}$$

This is a convex problem which looks for the sparse representation x_i^m for the i^{th} graph signal at each step where $i = 1, 2, ..., K_m$. The minimization on x_i^m corresponds to minimizing the equation in (3.13). This problem may solved by sparse coding algorithms such as the least absolute shrinkage and selection operator (LASSO) [39] or the orthogonal matching pursuit (OMP) algorithm [63]. When we solve this problem with OMP, we can directly impose a sparsity constraint on the sparse coefficient vectors x_i^m as $\|x_i^m\|_0 \leq T_0$. When we solve the problem with LASSO, although the sparsity of the coefficients cannot be adjusted exactly, one can tune the η_x parameter suitably in order to approximately bring the sparsity of the representation near a desired sparsity value T_0.

Step 2: Fix X^m, optimize ψ

$$\min_{\psi} \sum_{j=1}^{J} (\mu_j)^2 + \eta_s \sum_{j=1}^{J} (s_j - s_0)^2 + \eta_w \sum_{m=1}^{M} \|S^m Y^m - S^m D^m X^m\|_F^2 \tag{3.14}$$

The objective function here is a non-convex, smooth function of ψ. A local minimum of this function can be found with algorithms such as gradient descent [64], SQP [65], etc., which need the gradient of the objective function $f(\psi)$ of equation (3.14), to minimize the objective in the direction of the gradient.

We now describe the computation of the gradient of $f(\psi)$. The first two terms in (3.14) can be grouped as

$$f_{1,2}(\psi) = \Big(\underbrace{\begin{bmatrix} \mu_1 & \mu_2 & \cdots & \mu_J & s_1 & s_2 & \cdots & s_J \end{bmatrix}}_{\psi^T} - \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 & s_0 & s_0 & \cdots & s_0 \end{bmatrix}}_{a^T} \Big)$$
$$\underbrace{\begin{bmatrix} 1 & & & & & & & \\ & 1 & & & & & & \\ & & \ddots & & & & & \\ & & & 1 & & & & \\ & & & & \eta_s & & & \\ & & & & & \eta_s & & \\ & & & & & & \ddots & \\ & & & & & & & \eta_s \end{bmatrix}}_{N} \left(\underbrace{\begin{bmatrix} \mu_1 \\ \mu_2 \\ \vdots \\ \mu_J \\ s_1 \\ s_2 \\ \vdots \\ s_J \end{bmatrix}}_{\psi} - \underbrace{\begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ s_0 \\ s_0 \\ \vdots \\ s_0 \end{bmatrix}}_{a} \right) \tag{3.15}$$

$$\begin{aligned} f_{1,2}(\psi) &= \underbrace{(\psi - a)^T}_{1\times 2J} \underbrace{N}_{2J\times 2J} \underbrace{(\psi - a)}_{2J\times 1} \\ &= \psi^T N \psi - 2\psi^T N a + a^T N a \end{aligned} \tag{3.16}$$

The gradient of $f_{1,2}(\psi)$ with respect to ψ is

$$\frac{\partial f_{1,2}(\psi)}{\partial \psi} = 2N\psi - 2Na \tag{3.17}$$

In order to write the gradient of the third term in (3.14), first we must find the gradient

for each m:

$$
\begin{aligned}
f_3^m(\psi) &= \| \underbrace{S^m}_{L_m \times N_m} \underbrace{Y^m}_{N_m \times K_m} - \underbrace{S^m}_{L_m \times N_m} \underbrace{D^m}_{N_m \times JN_m} \underbrace{X^m}_{JN_m \times K_m} \|_F^2 \qquad (3.18)\\
&= \| \underbrace{\underbrace{S^m Y^m}_{L_m \times K_m}}_{A} - \underbrace{\underbrace{S^m U^m}_{L_m \times N_m}}_{B} \underbrace{G_m(\psi)}_{N_m \times JN_m} \underbrace{\underbrace{\begin{bmatrix} (U^m)^T & & & \\ & (U^m)^T & & \\ & & \ddots & \\ & & & (U^m)^T \end{bmatrix}}_{JN_m \times JN_m}}_{C} \underbrace{X^m}_{JN_m \times K_m} \|_F^2
\end{aligned}
$$

where $G_m(\psi) = \begin{bmatrix} \hat{g}_1(\Lambda_m) & \hat{g}_2(\Lambda_m) & \cdots & \hat{g}_J(\Lambda_m) \end{bmatrix}$. We can write $f_3^m(\psi)$ as

$$
\begin{aligned}
f_3^m(\psi) &= \|A - BG_m(\psi)C\|_F^2 \\
&= tr\Big((A - BG_m(\psi)C)^T (A - BG_m(\psi)C)\Big) \qquad (3.19)\\
&= tr(A^T A - A^T BG_m(\psi)C - C^T G_m(\psi)^T B^T A + C^T G_m(\psi)^T B^T BG_m(\psi)C).
\end{aligned}
$$

When we omit the constant terms, it becomes

$$
f_3^m(\psi) = tr(\underbrace{C^T G_m(\psi)^T B^T BG_m(\psi)C}_{H_1}) - 2tr(\underbrace{A^T BG_m(\psi)C}_{H_2}). \qquad (3.20)
$$

Taking the partial derivatives of $tr(H_1)$ and $tr(H_2)$ with respect to $G_m(\psi)$, we get

$$
\begin{aligned}
\frac{\partial tr(H_1)}{\partial G_m(\psi)} &= 2B^T BG_m(\psi)CC^T \qquad (3.21)\\
\frac{\partial tr(H_2)}{\partial G_m(\psi)} &= B^T AC^T. \qquad (3.22)
\end{aligned}
$$

Hence, the gradient of $f_3^m(\psi)$ is obtained as

$$
\underbrace{\frac{\partial f_3^m(\psi)}{\partial G_m(\psi)}}_{N_m \times JN_m} = 2 \underbrace{B^T B}_{N_m \times N_m} \underbrace{G_m(\psi)}_{N_m \times JN_m} \underbrace{CC^T}_{JN_m \times JN_m} - 2 \underbrace{B^T}_{N_m \times L_m} \underbrace{A}_{L_m \times K_m} \underbrace{C^T}_{K_m \times JN_m}. \qquad (3.23)
$$

Since the matrix

$$G_m(\psi) = \begin{bmatrix} \underbrace{\begin{bmatrix} \ddots & & 0 \\ & \ddots & \\ 0 & & \ddots \end{bmatrix}}_{\hat{g}_1(\Lambda_m)} & \underbrace{\begin{bmatrix} \ddots & & 0 \\ & \ddots & \\ 0 & & \ddots \end{bmatrix}}_{\hat{g}_2(\Lambda_m)} \cdots\cdots \underbrace{\begin{bmatrix} \ddots & & 0 \\ & \ddots & \\ 0 & & \ddots \end{bmatrix}}_{\hat{g}_J(\Lambda_m)} \end{bmatrix}_{N_m \times JN_m} \tag{3.24}$$

has only JN_m nonzero entries on the diagonal of each subdictionary $\hat{g}_1(\Lambda_m), \hat{g}_2(\Lambda_m), \ldots, \hat{g}_J(\Lambda_m)$, we form a vector from $\frac{\partial f_3^m(\psi)}{\partial G_m(\psi)}$ by extracting only the entries corresponding to these JN_m nonzero entries, which we denote as $\frac{\partial f_3^m}{\partial G_m^{JN_m}(\psi)} \in \mathbb{R}^{JN_m \times 1}$.

Equation (3.23) shows the gradient of $f_3^m(\psi)$ with respect to $G_m(\psi)$ for each m. In order to obtain the gradient of $f_3^m(\psi)$ with respect to ψ, we apply the chain rule.

$$\Big(\frac{\partial f_3^m(\psi)}{\partial \psi}\Big)^T = \Big(\frac{\partial f_3^m(\psi)}{\partial G_m^{JN_m}(\psi)}\Big)^T \frac{\partial G_m(\psi)}{\partial \psi} \tag{3.25}$$

Then we find the gradient of $G_m(\psi)$ with respect to ψ, which is $\frac{\partial G_m}{\partial \psi}$. We take the partial derivatives of each subdictionary $\hat{g}_j(\Lambda_m)$, where $j = 1, 2..., J$, with respect to all μ_j's and s_j's. We form vectors $\frac{\partial \hat{g}_j^m}{\partial \mu_j}$ and $\frac{\partial \hat{g}_j^m}{\partial s_j} \in \mathbb{R}^{N_m \times 1}$ by putting together the partial derivatives of the kernel functions $\hat{g}_j^m(\lambda)$ evaluated at the eigenvalues $\lambda_0^m, \ldots, \lambda_{N_m-1}^m$ of each graph. The partial derivatives of the kernel functions are computed as

$$\frac{\partial}{\partial \mu_j} e^{-\left(\frac{(\lambda^m - \mu_j)^2}{s_j^2}\right)} = \frac{2(\lambda^m - \mu_j)}{s_j^2} e^{-\left(\frac{(\lambda^m - \mu_j)^2}{s_j^2}\right)} \tag{3.26}$$

$$\frac{\partial}{\partial s_j} e^{-\left(\frac{(\lambda^m - \mu_j)^2}{s_j^2}\right)} = \frac{2(\lambda^m - \mu_j)^2}{s_j^3} e^{-\left(\frac{(\lambda^m - \mu_j)^2}{s_j^2}\right)} \tag{3.27}$$

$$\frac{\partial G_m}{\partial \psi} = \begin{bmatrix} \frac{\partial \hat{g}_1^m}{\partial \mu_1} & 0 & \cdots & 0 & \frac{\partial \hat{g}_1^m}{\partial s_1} & 0 & \cdots & 0 \\ 0 & \frac{\partial \hat{g}_2^m}{\partial \mu_2} & \cdots & 0 & 0 & \frac{\partial \hat{g}_2^m}{\partial s_2} & \cdots & 0 \\ 0 & 0 & \ddots & 0 & 0 & 0 & \ddots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & \frac{\partial \hat{g}_J^m}{\partial \mu_J} & 0 & 0 & \cdots & \frac{\partial \hat{g}_J^m}{\partial s_J} \end{bmatrix}_{JN_m \times 2J} . \tag{3.28}$$

Then we finally obtain the gradient of the third term as

$$\underbrace{\Big(\frac{\partial f_3^m(\psi)}{\partial \psi}\Big)^T}_{1\times 2J} = \underbrace{\Big(\frac{\partial f_3^m(\psi)}{\partial G_m^{JN_m}(\psi)}\Big)^T}_{1\times JN_m}\underbrace{\frac{\partial G_m(\psi)}{\partial \psi}}_{JN_m\times 2J}. \tag{3.29}$$

Finally, we get the overall gradient vector as

$$\begin{aligned}\frac{\partial f(\psi)}{\partial \psi} &= \frac{\partial f_{1,2}(\psi)}{\partial \psi} + \eta_w \sum_{m=1}^{M} \frac{\partial f_3^m(\psi)}{\partial \psi} \\ &= 2\underbrace{N}_{2J\times 2J}\underbrace{\psi}_{2J\times 1} - 2\underbrace{N}_{2J\times 2J}\underbrace{a}_{2J\times 1} + \eta_w \sum_{m=1}^{M} \underbrace{\frac{\partial f_3^m(\psi)}{\partial \psi}}_{2J\times 1}.\end{aligned} \tag{3.30}$$

We use this gradient vector $\frac{\partial f(\psi)}{\partial \psi}$ in the dictionary update stage of our algorithm to minimize $f(\psi)$.

Finally, we estimate the graph signals on each graph m from the learnt dictionaries and the sparse coefficients with the formula in (3.31). These graph signals $Y^m_{estimated}$ contain also the estimates of the incomplete entries.

$$Y^m_{estimated} = D^m X^m \tag{3.31}$$

Algorithm 1: Spectrally Concentrated Graph Dictionary Learning (SCGDL)

Initialization of dictionary D with random ψ;

while *1* **do**

Step 1: Fix D, update X coefficients with LASSO or OMP

for *m=1* $\rightarrow M$ **do**

for *i=1* $\rightarrow K_m$ **do**

$\min_{x_i^m} \eta_x \left\|x_i^m\right\|_1 + \eta_w \left\|S^m y_i^m - S^m D^m x_i^m\right\|_F^2$

end

end

Step 2: Fix X coefficients, update D (Gaussian kernel parameters ψ)

$\min_{\psi} \sum_{j=1}^{J} (\mu_j)^2 + \eta_s \sum_{j=1}^{J} (s_j - s_0)^2 + \eta_w \sum_{m=1}^{M} \left\|S^m Y^m - S^m D^m X^m\right\|_F^2$

if *maximum number of iterations reached* **then**

break;

end

end

Result: Sparse coefficients for each graph X^m and the Gaussian kernel parameters ψ

for *m=1* $\rightarrow M$ **do**

Estimate graph signals from the learnt dictionary and the sparse coefficients:

$Y^m_{estimated} = D^m X^m$

end

3.4 Complexity Analysis of the Algorithm

In this section, the complexity of the proposed method is analyzed. First, we assume that the graph weight matrix $\mathcal{W}_m$ is known. Then, the time complexity of calculating the graph Laplacian $\mathcal{L}^m$ is $O(N_m^2)$ and the complexity of the eigenvalue decomposition of $\mathcal{L}^m$ is $O(N_m^3)$. We examine the time complexity of the proposed algorithm with two steps, by calculating the complexity of the sparse coding stage and the dictionary update stage.

In the sparse coding stage when LASSO algorithm is used, the sparse coefficients can be computed with $O(K_m L_m J^2 N_m^2 + K_m J^3 N_m^3)$ operations for a single graph $\mathcal{G}_m$ [66]. Approximating the parameters for the individual graphs roughly as $N_m \approx N$, $K_m \approx K$ and $L_m \approx L$, the overall complexity is expected to be around $O(MKLJ^2N^2 + MKJ^3N^3)$ for all M graphs. When the OMP algorithm is used in the sparse coding stage, we must consider the time complexity of Gram-Schmidt (G.S.) orthogonalization and projections of the signals. The time complexity of G.S. orthogonalization for a graph signal is $O(L_m J^2 N_m^2)$, we have K_m graph signals and the time complexity of doing G.S. orthogonalization for K_m graph signals will be $O(K_m L_m J^2 N_m^2)$. We need to calculate the time complexity of projections of those graph signals, to project a graph signal $O(L_m J N_m)$ operations are needed. When we project K_m graph signals, the complexity will be $O(K_m L_m J N_m)$. The total time complexity of the OMP algorithm is thus $O(K_m L_m J N_m + K_m L_m J^2 N_m^2)$. This complexity reduces to $O(K_m L_m J^2 N_m^2)$ by writing only the higher order terms. We can roughly express this complexity as $O(MKLJ^2N^2)$ for all M graphs by approximating the parameters for the individual graphs as above.

In the dictionary update stage, firstly, we analyze the time complexity of the function in equation (3.14) and then its gradient. The equation in (3.14) can be computed with $O(J + K_m L_m + K_m J N_m^2)$ operations. Since J is smaller than N_m, this complexity can be also written as $O(K_m L_m + K_m J N_m^2)$. The time complexity of the function in equation (3.14) will be $O(MKL + MKJN^2)$ for all M graphs by approximating the parameters of K_m, L_m and N_m.

In order to observe the time complexity of the gradient of the equation (3.14), we consider the first two terms in equation (3.14) whose gradient is given in equation (3.17). The time complexity of calculating this gradient is $O(J)$. For the time complexity of the gradient of the third term in equation (3.14), we must calculate the time complexities of equations (3.23), (3.28) and (3.29). We begin with the time complexity of the first term in equation (3.23) which will be calculated with $O(L_m N_m^2 + J N_m + K_m J^2 N_m^2 + K_m J^2 N_m^2 + J N_m^3 + J^2 N_m^3)$ operations. The second term of (3.23) can be computed with $O(K_m L_m N_m + K_m J N_m^2)$ operations. The total time complexity of equation (3.23) reduces to $O(L_m N_m^2 + K_m J^2 N_m^2 + J^2 N_m^3 + K_m L_m N_m)$ by writing only the higher order terms. Then, the time complexities of the equations

(3.28) and (3.29) can be calculated with $O(JN_m)$ and $O(J^2N_m)$ operations, respectively. The total time complexity of the calculating the gradient of the function is $O(J + L_mN_m^2 + K_mJ^2N_m^2 + J^2N_m^3 + K_mL_mN_m + JN_m + J^2N_m)$. When we write this complexity with only the higher order terms, it will be $O(L_mN_m^2 + K_mJ^2N_m^2 + J^2N_m^3 + K_mL_mN_m)$. Approximating the parameters for the individual graphs roughly as $N_m \approx N$, $K_m \approx K$ and $L_m \approx L$, the overall complexity is expected to be around $O(MLN^2 + MKJ^2N^2 + MJ^2N^3 + MKLN)$ for all M graphs.

Putting together the complexities of the sparse coding and the dictionary update stages, we thus get the overall complexity of the proposed SCGDL algorithm as $O(MKLJ^2N^2 + MKJ^3N^3 + MKL + MKJN^2 + MLN^2 + MKJ^2N^2 + MJ^2N^3 + MKLN)$ when LASSO algortihm is used in the sparse coding stage. The complexity reduces to $O(MKLJ^2N^2 + MKJ^3N^3)$ by writing only the higher order terms. When the OMP algorithm is used in the sparse coding stage, we get the overall complexity of the proposed SCGDL algorithm as $O(MKLJ^2N^2 + MKL + MKJN^2 + MLN^2 + MKJ^2N^2 + MJ^2N^3 + MKLN)$. This complexity reduces to $O(MKLJ^2N^2 + MJ^2N^3)$ by writing only the higher order terms.

CHAPTER 4

EXPERIMENTAL RESULTS

In this chapter, we evaluate the performance of our algorithm with comparative experiments on a synthetic graph signal set.

4.1 Synthetic Data Generation

In order to generate synthetic data, firstly, graphs must be generated. We create $M = 2$ graphs by first randomly initializing the graph nodes, whose locations are i.i.d. and drawn from a uniform distribution. After calculating the distances between these randomly positioned nodes, we connect each pair of nodes with an edge and assign the edge weights as a Gaussian function of the distance between the nodes. Then we refine the graph topology in order to obtain an Erdős–Rényi graph. This is done by keeping the edges with probability p_{erdos} and removing them with probability $1 - p_{erdos}$ independently of each other, where p_{erdos} is a predetermined probability parameter. When an edge is removed, its weight is changed to $W_{ij} = 0$. Determining the weight matrices for the two graphs in this way, the normalized graph Laplacians for the two graphs are defined as

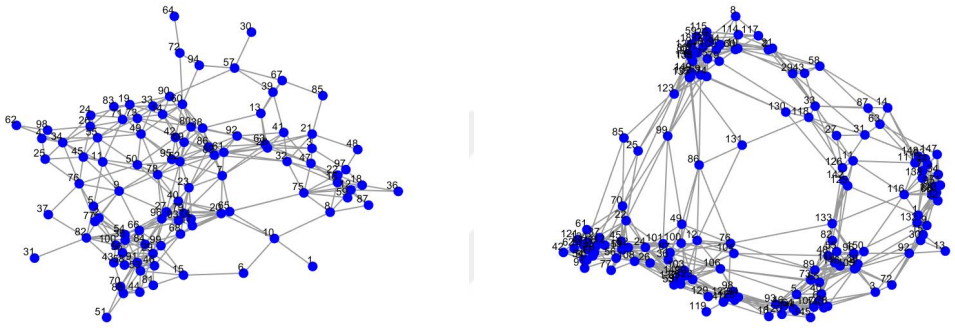
$$\mathcal{L}_m = \mathcal{I} - {\mathcal{D}_m}^{-1/2} \mathcal{W}_m {\mathcal{D}_m}^{-1/2} \tag{4.1}$$

where $\mathcal{W}_m$ is the weight matrix, $\mathcal{D}_m$ is the degree matrix and $\mathcal{L}_m$ is the graph Laplacian which is a real symmetric matrix with a complete set of orthonormal eigenvectors and nonnegative eigenvalues, for $m = 1, 2$. The eigenvectors of the graph Laplacian $\mathcal{L}_m$ are denoted by $U^m = [U_0^m, U_1^m, \ldots, U_{N_m}^m]$, and its eigenvalues $0 = \lambda_0^m < \lambda_1^m \leq$

$\lambda_2^m \leq ... \leq \lambda_{(N_m-1)}^m \leq 2$ are often represented with the diagonal matrix:

$$\Lambda^m := \begin{bmatrix} \lambda_0^m & & 0 \\ & \ddots & \\ 0 & & \lambda_{N_m-1}^m \end{bmatrix} \tag{4.2}$$

We generate two graphs ($M = 2$) whose topologies are illustrated in Figure(4.1). We generate $\mathcal{G}_1$ with $N_1 = 100$ nodes and $\mathcal{G}_2$ with $N_2 = 150$ nodes.



(a) (b)

Figure 4.1: Generated graphs $\mathcal{G}_1$ (a) and $\mathcal{G}_2$ (b) for the simulations.

In order to generate a graph signal y_i^m on graph $\mathcal{G}_m$, we first generate a dictionary from subdictionaries obtained from the graph kernels $\hat{g}_j(\lambda)$ in (4.3). We begin with specifying the mean μ_j and the scale s_j values of the Gaussian kernel parameters to construct our parameter vector ψ_{syn}. In this thesis, we use four graph kernels ($J = 4$) with parameters

$$\theta_j = (\mu_j, \sigma_j)\ ,\ \hat{g}_j(\lambda) = e^{-\left(\frac{\left\|\lambda^m - \mu_j\right\|^2}{s_j^2}\right)} \tag{4.3}$$

where the random variables $\begin{cases} \mu_j \sim \mathcal{N}(0, 0.6) \\ s_j \sim \mathcal{N}(0.25, 0.05) \end{cases}$ have a joint normal density with

mean and covariance $m_{\underline{\theta}} = \begin{bmatrix} 0 \\ 0.25 \end{bmatrix}, \sum_{\underline{\theta}} = \begin{bmatrix} 0.6 & 0 \\ 0 & 0.05 \end{bmatrix}$.

The mean and the scale values of the Gaussian graph kernels are sampled independently from normal distributions with variances $\sigma_\mu^2 = 0.6$, $\sigma_s^2 = 0.05$ and s_0 is a vector with $J = 4$ elements $s_0 = [0.25\ 0.25\ 0.25\ 0.25]$. In an example run of our simulations, the ground truth Gaussian kernel parameters in the ψ_{syn} vector have been drawn as

$$\psi_{syn}^T = \begin{bmatrix} 0.3694 & 0.4454 & 0.6369 & 0.8950 & 0.2874 & 0.2404 & 0.2944 & 0.2118 \end{bmatrix}.$$

We fix the ground truth kernel parameter vector in this way throughout the experiments in this chapter. The illustration of the resulting Gaussian graph kernels is given in Figure (4.2).

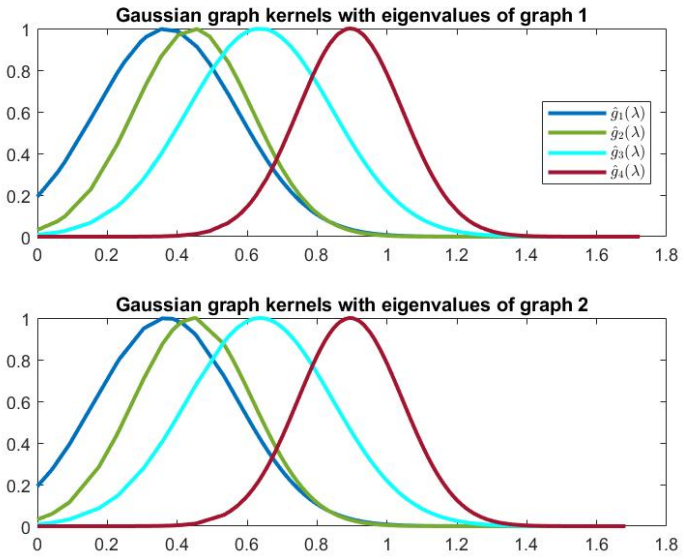


Figure 4.2: Illustration of the Gaussian kernels generated with the ψ_{syn} vector as a function of the eigenvalues of the two synthetic graphs

The kernel $\hat{g}_j(\lambda)$ generates a subdictionary D_j^m when applied to the eigenvalues of the graph Laplacian on graph m as

$$D_j^m = U^m \hat{g}_j(\Lambda^m)(U^m)^T \in \mathbb{R}^{N_m \times N_m} \quad \text{where} \quad \mathcal{L}_m = U^m \Lambda^m (U^m)^T \tag{4.4}$$

A structured dictionary is then obtained on each graph as in equation (4.6), which is the concatenation of $J = 4$ subdictionaries.

$$D^m = \begin{bmatrix} D_1^m & D_2^m & D_3^m & D_4^m \end{bmatrix} \tag{4.5}$$

As mentioned in Section 3.1, the entries of the sparse coefficients x_i^m of the graph signals are sampled from the Laplace distribution for the desired sparsity value and the noise signals are sampled i.i.d. from a normal distribution. The graph signals are then formed as

$$y_i^m = \begin{bmatrix} D_1^m & D_2^m & D_3^m & D_4^m \end{bmatrix} x_i^m + w_i^m = D^m x_i^m + w_i^m \tag{4.6}$$

where $i = 1, 2, \ldots, K_m$. Two graph signal sets containing respectively $K_1 = 500$ and $K_2 = 400$ signals are generated on the two graphs as in equation (4.6). The graph signals on the two graphs are represented with the Y^1 and Y^2 matrices as follows

$$Y^1 = \begin{bmatrix} \vdots & \cdots & \vdots \\ y_1^1 & \cdots & y_{500}^1 \\ \vdots & \cdots & \vdots \end{bmatrix} \in \mathbb{R}^{100 \times 500} \quad , \quad Y^2 = \begin{bmatrix} \vdots & \cdots & \vdots \\ y_1^2 & \cdots & y_{400}^2 \\ \vdots & \cdots & \vdots \end{bmatrix} \in \mathbb{R}^{150 \times 400}$$

4.2 Experimental Results

In the experiments, we investigate the performance of the proposed algorithm in comparison with two reference methods; namely, the graph-based SSL algorithm [4] and the graph dictionaries obtained with the SGWT [58]. As discussed in Chapter 2, the SSL method estimates the missing entries of the graph signals based on smoothness constraints, while the SGWT method computes fixed graph wavelet dictionaries that efficiently cover the graph spectrum. Four different wavelet kernel types are tested for the SGWT method, which are the Abspline kernel, Mexican hat kernel, Meyer kernel and the Tight-frame kernel [58]. Firstly, we examine the effect of the ratio of missing observations and the effect of noise on the performance of the algorithms. Secondly, we study the effect of the number of Gaussian graph kernels generating the graph dictionaries. Thirdly, we examine the effect of the sparsity constraint value in the sparse

coding stage of the algorithm. Lastly, we compare two possible dictionary learning strategies within the framework of our algorithm, where the dictionaries are learnt jointly on the two graphs in the first strategy and individually on each graph in the second strategy. Lastly, we study the sensitivity analysis of the proposed algorithm to the algorithm parameters.

As an evaluation metric, we use the mean-squared error (MSE) between the ground truth signal values and their estimates, which is given in the equation below.

$$MSE_Y = \frac{\left\| Y^m_{unknown} - Y^m_{estimated,unknown} \right\|_F^2}{\left\| Y^m_{unknown} \right\|_F^2}. \tag{4.7}$$

Here $Y^m_{unknown}$ is the matrix containing the ground truth values of the missing observations of the graph signals and $Y^m_{estimated,unknown}$ is the matrix containing their estimates on each graph m.

4.2.1 Comparative Evaluation of Algorithm Performance

In an example application where the graphs model sensor networks, the i-th row of the graph signal matrix represents all sensor measurements taken by the i-th sensor (at node i) and different columns of the matrix represent different measurements. When the sensor on node i is out of order, we do not observe any measurement from that node, which means that the measurements on node i are missing. Motivated by this example application, we consider an experimental setup where 10%, 20%, 30%, 40%, 50% of the sensors are out of order and their measurements are missing. Randomly selecting the nodes with missing measurements, we form the graph signal data set by deleting the corresponding entries of the synthetic graph signals. One important point about this setup is that while the measurements on these nodes are missing, the nodes themselves are still in the graph topology.

In order to analyze the effect of the ratio of missing observations, graph signals are generated as in equation (4.6), however without noise. In the sparse coding stage of the algorithm, we use both the LASSO and the OMP algorithms. We select the sparsity constraint parameter as $T_0 = 40$. When OMP is used, the sparsity parameter can be directly set to this desired value. However when LASSO is used in the sparse coding stage, we can not assign the sparsity constraint directly, instead we keep this value

within a certain range around $T_0 = 40$ by adjusting the η_w and η_x parameters. We run our algorithm for a certain number of repetitions with different random initializations of the initial solution for the parameter vector ψ in the dictionary update step of the algorithm and different random selections of the nodes where the observations are missing, in each repetition of the simulations. The reported results are obtained as the average of 30 repetitions. The mean square error of the estimation of the missing observations can be seen in Figure (4.3) and Figure (4.4).

The proposed algorithm outperforms the graph-based SSL and SGWT algorithms when LASSO is used in the sparse coding stage. However, when the OMP algorithm is used in the sparse coding stage, in the case where 50% of the observations are missing, the proposed algorithm can not outperform the graph-based SSL algorithm for the first synthetic graph $\mathcal{G}_1$. As the ratio of missing measurements increases, the error generally increases but especially after 40% it increases more rapidly.

Another important conclusion from this experiment is that the performance of the SGWT algorithm does not very well compare to the others. SGWT generates fixed dictionaries that are adapted to the graph topology but not particularly to the signal set at hand. Meanwhile, our algorithm learns the dictionary by taking into account both the graph signals and the graph topology, which explains the performance gain achieved in comparison to SGWT.

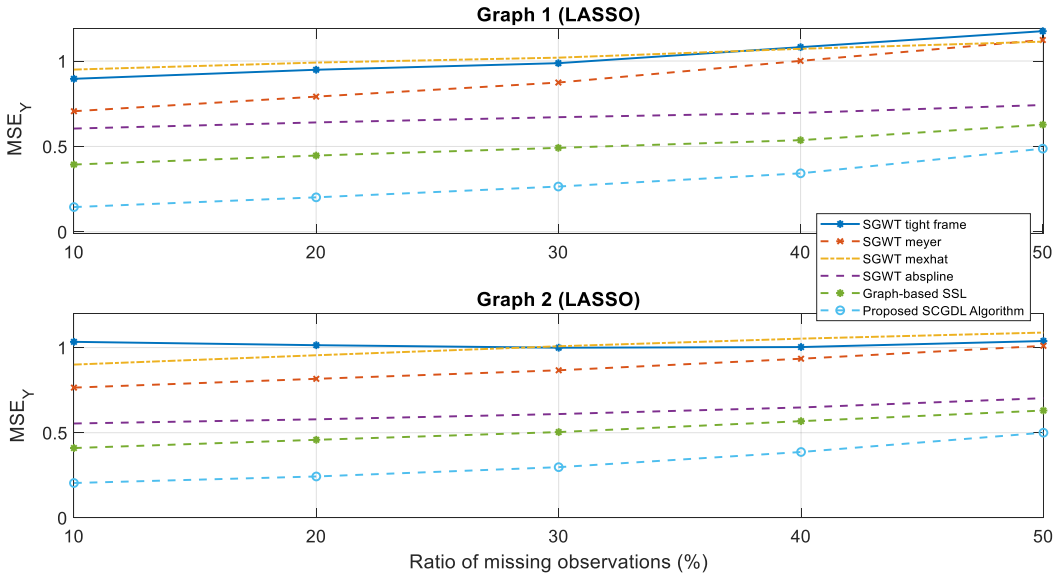


Figure 4.3: MSE values for different rates of missing measurements when LASSO is used in the sparse coding stage.

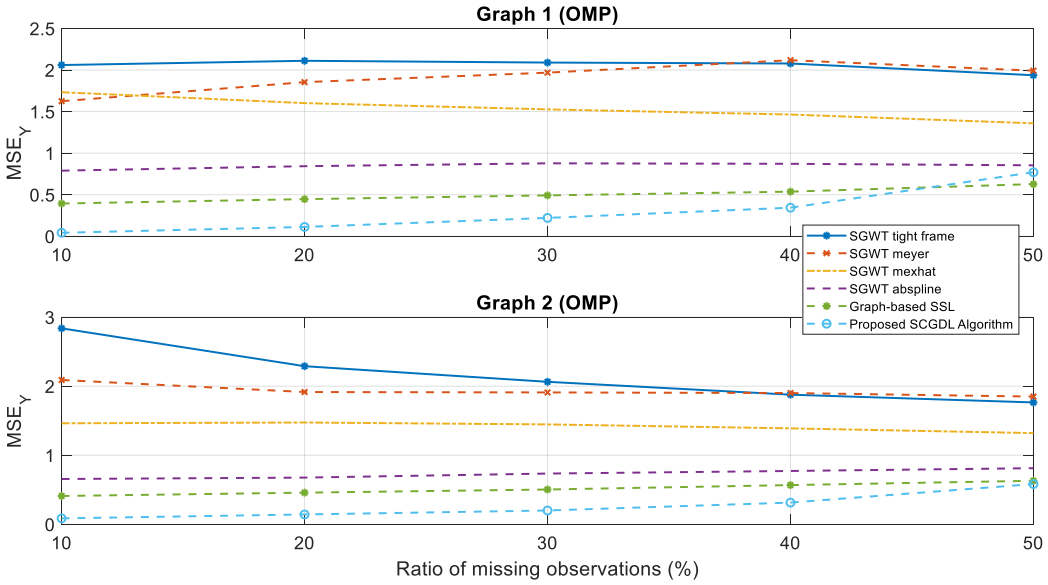


Figure 4.4: MSE values for different rates of missing measurements when OMP is used in the sparse coding stage.

Next, in order to examine the effect of the noise level on the algorithm performance, we generate graph signals with different noise levels in equation (4.6). The noise

signals are sampled from normal distributions with different noise variances. We evaluate the performance of our algorithm and the other reference algorithms. We set the sparsity constraint as $T_0 = 40$ and run our algorithm for 30 repetitions and average the results. We fix the ratio of missing observations to 20%. As one can see in Figures (4.5) and (4.6), the performance of our algorithm for $\mathcal{G}_1$ and $\mathcal{G}_2$ are not as good as the graph-based SSL method at the highest noise level, however when the noise level falls to a reasonable range (corresponding to around $6 - 9$ dB) the proposed SCGDL algorithm outperforms the reference algorithms. This can be explained in the way that the smoothing effect of SSL is beneficial at very high noise levels. At high noise levels, our algorithm may tend to shift the learnt kernels towards the high-frequency range of the spectrum, trying to match it to noise which affects its performance in a negative way. However, if the noise level remains at a reasonable range, the kernels learned by our algorithm can fit the clean signal components instead of the noise component.

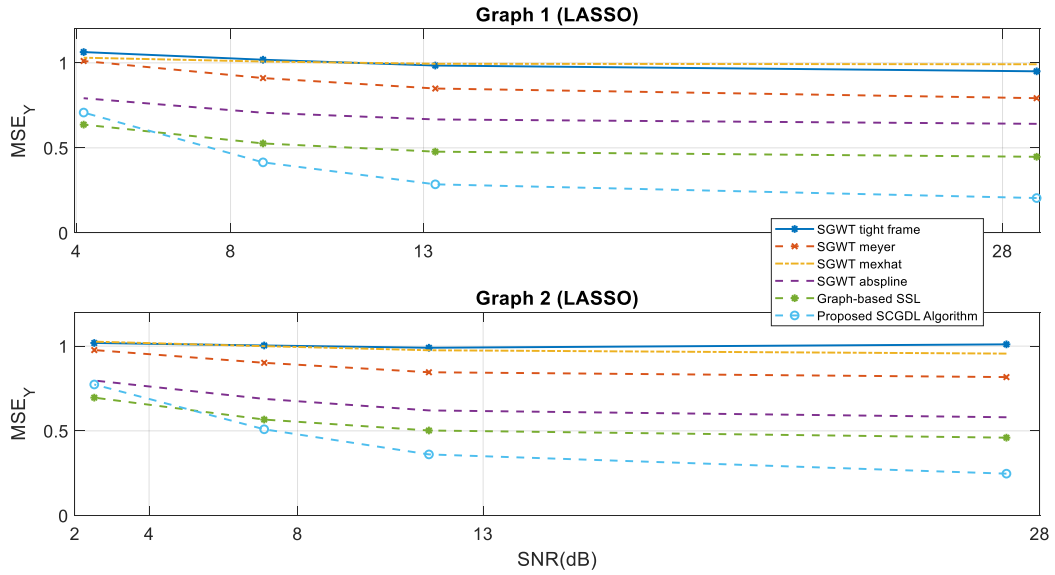


Figure 4.5: MSE values for different SNR (dB) values when LASSO is used in the sparse coding stage.

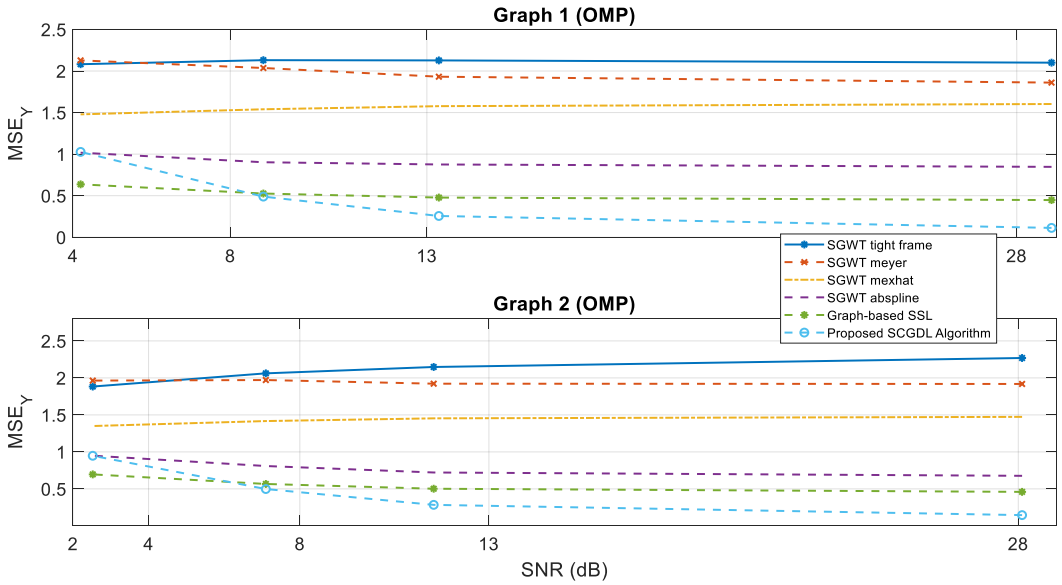


Figure 4.6: MSE values for different SNR (dB) values when OMP is used in the sparse coding stage.

4.2.2 Learning the Dictionary with Different Number of Graph Kernels

An important parameter that may affect the performance of our algorithm is the number of Gaussian kernels used in the generation of our dictionaries.

In this experiment, we examine the effect of learning the dictionary with different numbers of graph kernels. While we generate the ground truth signals with $J = 4$ kernels, at the dictionary update step, we select fewer kernels ($J = 2$ and $J = 3$) to generate the dictionary and calculate the resulting MSE values. We fix the missing observation ratio to 20%. We run our algorithm for 30 random repetitions with noiseless graph signals.

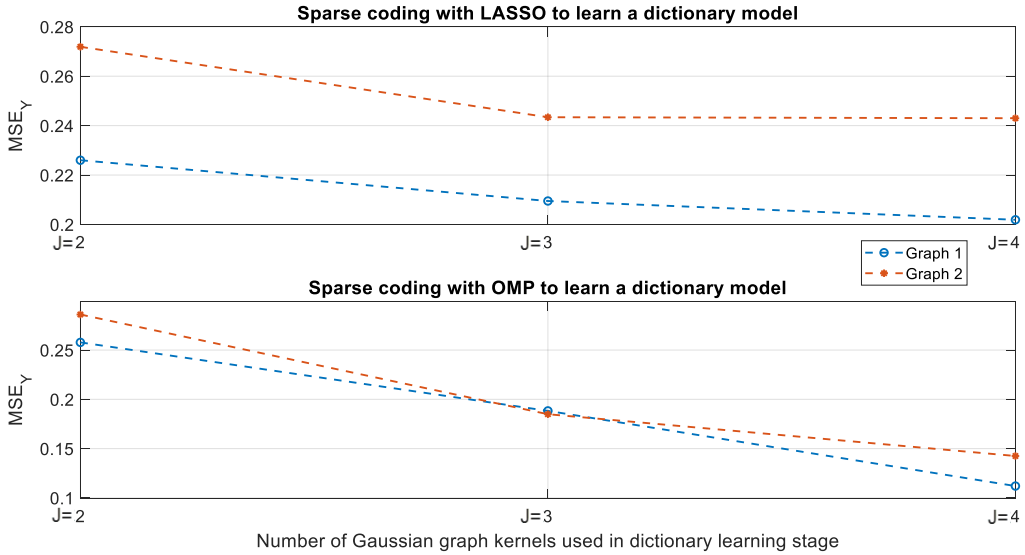


Figure 4.7: Comparison of the MSE values for different number of Gaussian kernels used in the dictionaries.

As one can see in Figure (4.7), the proposed algorithm attains its best performance for both $\mathcal{G}_1$ and $\mathcal{G}_2$ when the number of Gaussian kernels is chosen $J = 4$ in the dictionary learning stage. Remembering that the signals are obtained from dictionaries generated with $J = 4$ graph kernels, these results can be interpreted in the following way: The ground truth value $J = 4$ may be regarded in the way that the spectrum of the signals has a certain complex structure. When we choose J to be smaller and learn fewer kernels, the dictionary model computed with this J value does not fit well enough to the complex structure of the graph signals, therefore it can not model the signals well enough. One can understand from this experiment that when learning dictionaries, it is critical to choose the number of graph kernels suitably so that the structure of the signal set at hand can be represented sufficiently well.

4.2.3 The Effect of the Sparsity Constraints on the Performance

We now examine the performance of the algorithm under different sparsity constraints where 20% of the observations are missing. For LASSO, we can not strictly fix the sparsity value, but we determine the parameters η_w and η_x so as to approach

the sparsity values which are given in Figure (4.8). We run our algorithm for 30 repetitions for noiseless graph signals.

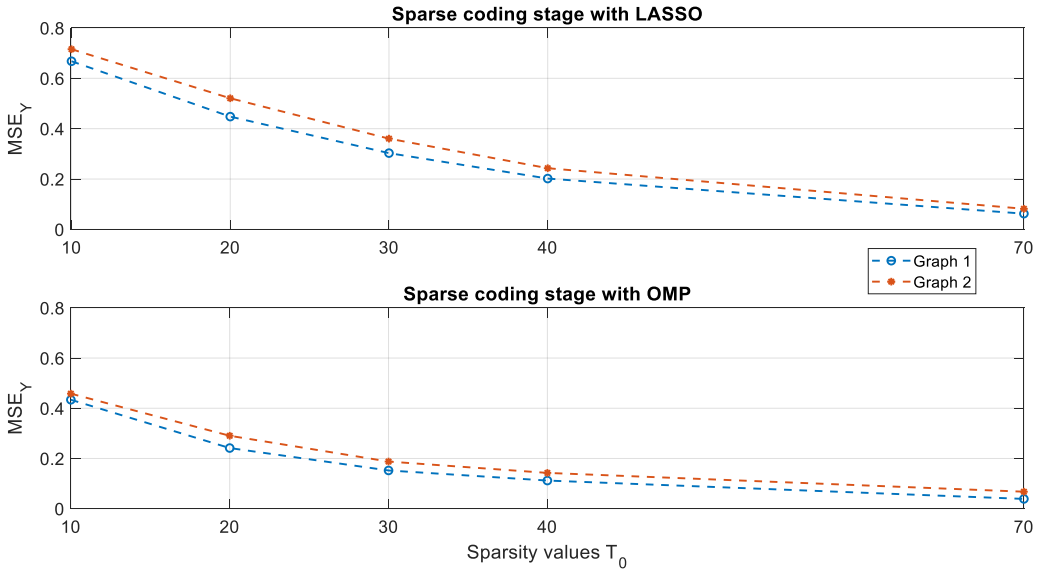


Figure 4.8: Comparison of the MSE values for different sparsity values T_0 used in the sparse coding stage.

As the sparsity constraint value T_0 gets higher, in the representation of the graph signals we use more atoms, which makes the graph signal representation less sparse. In Figure (4.8), we observe that the MSE decreases monotonically as the signals are represented with more atoms. This may seem contradictory to the expectation to observe an optimal value for the sparsity parameter T_0. Nevertheless, this can be explained with the reasoning that in such missing data scenarios, using more atoms may provide robustness in the graph signal representation.

4.2.4 Comparison of Dictionary Learning on Single and Multiple Graphs

In the proposed algorithm, we study the estimation of partially observed multiple graph signals by learning spectrally concentrated graph kernels. Our algorithm is generic in the sense that the dictionary-generating Gaussian kernels can be learnt from signal sets defined over more than one graph. If the graphs share similar features, the usage of multiple graphs may be helpful. In this experiment, we examine whether it is

more advantageous to learn common dictionaries jointly over multiple graphs or learn them individually over each graph. We compare these two schemes under the settings where 10%, 20%, 30%, 40%, 50%, 70%, 90% of the observations are missing. We run our algorithm for 30 repetitions for noiseless graph signals. In the sparse coding stage, the LASSO algorithm is used with a sparsity constraint of around $T_0 = 40$.

	MG	MG	SG	SG
Ratio of missing observations	$\mathcal{G}_1$	$\mathcal{G}_2$	$\mathcal{G}_1$	$\mathcal{G}_2$
10%	0.14	0.20	1.66	0.11
20%	0.20	0.24	0.07	0.13
30%	0.26	0.30	0.11	0.14
40%	0.34	0.39	0.19	0.25
50%	0.49	0.50	0.29	0.35
70%	0.91	0.88	0.67	0.80
90%	0.97	0.99	0.91	0.98

Table 4.1: Comparison of the MSE values for dictionary learning on multiple graphs (MG) and a single graph (SG)

In general, it is inferred from Table 4.1 that individual dictionary learning on single graphs gives a lower MSE than joint dictionary learning on multiple graphs. This may result from the fact that the two graph topologies used in these experiments may not be similar enough. However, one can see that the results of dictionary learning on multiple graphs (MG) are slightly more "regulated" than dictionary learning on single graphs (SG). For example, SG gives a very high error of 1.6562 in the first graph at 10% missing data rate. This type of outlier behavior seems to be avoided in the MG setting, where the two graphs are used together when learning the dictionary.

4.2.5 Algorithm Parameters Sensitivity Analysis

In the objective function, some algorithm parameters such as η_s, η_w and η_x are used to control the contribution of the corresponding terms. In order to analyze the sensitivity

of the algorithm performance to the algorithm parameters, we fix the values of some parameters, then examine the effect of the others on the algorithm performance. For this purpose, we run our algorithm for 10 repetitions. We first fix $\eta_w = 10^4$ and $\eta_x = 430$ to keep the sparsity parameter around the value 40, and examine the effect of the η_s parameter on the algorithm performance. Table 4.2 shows the MSE_Y values for different values of η_s for $\mathcal{G}_1$ and $\mathcal{G}_2$.

η_s	$\mathcal{G}_1$	$\mathcal{G}_2$
10^5	0.88	0.90
10^6	0.72	0.81
10^7	0.27	0.44
10^8	0.17	0.27
10^9	0.16	0.25

Table 4.2: Comparison of the MSE values for different values of η_s for $\mathcal{G}_1$ and $\mathcal{G}_2$

The parameter η_s gives better MSE_Y values when it takes high values. Increasing the value of the parameter η_s improves the performance. This shows that allowing the spectral kernels to deviate from their narrowband structure, which happens at relatively small η_s values, has a negative impact on the performance.

η_w	η_x	$\mathcal{G}_1$	$\mathcal{G}_2$
10^2	4.3	0.16	0.25
10^3	43	0.16	0.25
10^4	430	0.17	0.27
10^5	4300	0.31	0.48
10^6	43000	0.71	0.80

Table 4.3: Comparison of the MSE values for different values of the η_w-η_x pair.

Next, we fix $\eta_s = 10^8$, and examine the effects of the η_w and η_x parameters on the algorithm performance. Due to the objective function used in the sparse coding stage, the ratio between the η_w and η_x parameters must be constant to obtain the desired sparsity value which is taken as around $T_0 = 40$ for this experiment.

As one can see from Table 4.3, when the values of the η_w-η_x pair increase, the MSE_Y values on graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ also increase. The η_x and η_w parameters weigh the sparsity and the signal approximation terms in the objective function. Increasing these parameters too much decreases the impact of the other two terms in the objective that define the structure of the spectral Gaussian kernels, hence, affects the performance negatively.

CHAPTER 5

CONCLUSION

In this thesis, we proposed a method that estimates partially observed multi-graph signals by learning spectrally concentrated graph kernels. Our study is motivated by the observation that in many practical scenarios graph signals may contain components exhibiting local and isolated behavior in certain graph regions. In these cases, it is important to study how these signal components are distributed across the frequency spectrum. Unlike many state-of-the-art methods making the "smoothly changing graph signal" assumption, we aimed to identify the regions in the graph spectrum with rather band-pass and high-pass characteristics. We modelled the graph spectrum via Gaussian graph kernels by fitting their mean and scale parameters to the observed graph signals in order to capture this behaviour. We aimed to handle the difficulties arising from the incompleteness of the observations by using multiple graphs, under the assumption that the signals observed on them have similar features. We demonstrated the efficacy of our algorithm on synthetic data sets in various settings. From a given data set of graph signals exhibiting local variations on the graph leading to band-pass and high-pass components, we showed that our algorithm can successully capture these frequency components by learning spectrally concentrated graph dictionaries.

In the experiments, we used the mean-squared error (MSE) between the ground truth signal values and their estimates as an evaluation metric. We compared the MSE values of our method to those of the reference methods graph-based SSL and SGWT dictionaries. In the first experiment, we investigated the effect of the ratio of missing observations and the effect of noise on the performance of the algorithms. As the ratio of missing observations increases, the error increases for all algorithms. Except the

case where 50% of the observations are missing for the first graph when using OMP in the sparse coding stage, the proposed SCGDL method outperforms the reference methods. In the noise sensitivity analysis, at very high noise levels (around $1-5$ dB) the performance of our algorithm has been observed to be not as good as that of graph-based SSL. When the noise level falls to a reasonable range, the proposed algorithm outperforms the other algorithms.

In the second experiment, we studied the effect of the number of Gaussian graph kernels used in the dictionaries. We noticed that the performance of the algorithm decreased when the dictionary was generated with fewer Gaussian graph kernels, because it does not fit well enough to the complex structure of the graph signals, hence it can not model the signals well enough. This points to the importance of choosing and learning a sufficiently rich dictionary model that can successfully capture the structure of the signal set at hand.

In the third experiment, we examined the effect of the sparsity constraint value in the sparse coding stage of the algorithm. We noticed that when the signals are represented with a higher number of atoms, the performance is better. This can be explained in the way that in such missing data scenarios, using more atoms may provide robustness in the graph signal representation.

In the fourth experiment, we compared the performances of dictionary learning on multiple graphs and a single graph. The results showed that while learning the dictionaries individually on the graphs may be preferable in some settings especially if the graph topologies are not sufficiently similar, the joint learning of a common dictionary may have a "regulating" effect that may help reduce the effects of some outlier cases. Finally, in the last experiment, we examined the sensitivity of the proposed SCGDL algorithm to the algorithm parameters.

As future work, the proposed idea of estimating incomplete graph data from signal sets on multiple graphs can be tested on more than two graphs. Another important future direction is to evaluate the performance of our algorithm on real data sets. It is important to examine how well the graph signals observed in real applications adhere to our proposed graph signal model, and how the performance of our algorithm gets affected by the deviation of these signals from our model. As another future work,

our algorithm can be adapted to a more general scenario that may arise in certain applications: In this thesis, we have focused on a setting where a sensor located on a node is out of order for all signal observations; hence the signal value observed on a node is either present or missing in all of the input graph signals. Differently, a setup can be considered in which different sensors are broken for different observations of the graph signals, e.g., the i^{th} sensor may be out of order for certain measurements or observation instances, while it may be fixed and operational for others. The extension of our algorithm to cover such scenarios may be a useful future study.

REFERENCES

[1] Aliaksei Sandryhaila and Jose MF Moura. Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure. *IEEE Signal Processing Magazine*, 31(5):80–90, 2014.

[2] Antonio Ortega, Pascal Frossard, Jelena Kovačević, José MF Moura, and Pierre Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, 2018.

[3] David I Shuman, Sunil K Narang, Pascal Frossard, Antonio Ortega, and Pierre Vandergheynst. The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains. *IEEE signal processing magazine*, 30(3):83–98, 2013.

[4] Xiaojin Zhu, John Lafferty, and Ronald Rosenfeld. *Semi-supervised learning with graphs*. PhD thesis, Carnegie Mellon University, language technologies institute, school of . . . , 2005.

[5] Avrim Blum and Shuchi Chawla. Learning from labeled and unlabeled data using graph mincuts. 2001.

[6] Avrim Blum, John Lafferty, Mugizi Robert Rwebangira, and Rajashekar Reddy. Semi-supervised learning using randomized mincuts. In *Proceedings of the twenty-first international conference on Machine learning*, page 13, 2004.

[7] Xiaojin Zhu and Zoubin Ghahramani. Towards semi-supervised classification with markov random fields. 2002.

[8] Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pages 912–919, 2003.

[9] Dengyong Zhou, Olivier Bousquet, Thomas Navin Lal, Jason Weston, and Bern-

hard Schölkopf. Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328, 2004.

[10] Misha Belkin, Partha Niyogi, and Vikas Sindhwani. On manifold regularization. In *AISTATS*, volume 1, 2005.

[11] Dorina Thanou and Pascal Frossard. Multi-graph learning of spectral graph dictionaries. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3397–3401. IEEE, 2015.

[12] Xuan Zhang, Xiaowen Dong, and Pascal Frossard. Learning of structured graph dictionaries. In *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3373–3376. IEEE, 2012.

[13] Dorina Thanou, David I Shuman, and Pascal Frossard. Parametric dictionary learning for graph signals. In *2013 IEEE Global Conference on Signal and Information Processing*, pages 487–490. IEEE, 2013.

[14] Dorina Thanou, David I Shuman, and Pascal Frossard. Learning parametric dictionaries for signals on graphs. *IEEE Transactions on Signal Processing*, 62(15):3849–3862, 2014.

[15] Hermina Petric Maretic, Dorina Thanou, and Pascal Frossard. Graph learning under sparsity priors. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6523–6527. Ieee, 2017.

[16] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[17] Xiaojin Jerry Zhu. Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences, 2005.

[18] Charles J Geyer. Practical markov chain monte carlo. *Statistical science*, pages 473–483, 1992.

[19] Mikhail Belkin, Irina Matveeva, and Partha Niyogi. Regularization and semi-supervised learning on large graphs. In *International Conference on Computational Learning Theory*, pages 624–638. Springer, 2004.

[20] Satoshi Furutani, Toshiki Shibahara, Mitsuaki Akiyama, Kunio Hato, and Masaki Aida. Graph signal processing for directed graphs based on the hermitian laplacian. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 447–463. Springer, 2019.

[21] Leo J Grady and Jonathan R Polimeni. Introduction to discrete calculus. In *Discrete Calculus*, pages 13–89. Springer, 2010.

[22] Matthias Hein, Jean-Yves Audibert, and Ulrike Von Luxburg. From graphs to manifolds–weak and strong pointwise consistency of graph laplacians. In *International Conference on Computational Learning Theory*, pages 470–485. Springer, 2005.

[23] Mario Coutino, Elvin Isufi, and Geert Leus. Advances in distributed graph filtering. *IEEE Transactions on Signal Processing*, 67(9):2320–2333, 2019.

[24] Aliaksei Sandryhaila and Jose MF Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Transactions on Signal Processing*, 62(12):3042–3054, 2014.

[25] David I Shuman, Benjamin Ricaud, and Pierre Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[26] Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7(Nov):2399–2434, 2006.

[27] Jeremy Ma, Weiyu Huang, Santiago Segarra, and Alejandro Ribeiro. Diffusion filtering of graph signals and its use in recommendation systems. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4563–4567. IEEE, 2016.

[28] Benjamin Girault, Paulo Gonçalves, Eric Fleury, and Arashpreet Singh Mor. Semi-supervised learning for graph to signal mapping: A graph signal wiener filter interpretation. In *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1115–1119. IEEE, 2014.

[29] Elvin Isufi, Paolo Di Lorenzo, Paolo Banelli, and Geert Leus. Distributed wiener-based reconstruction of graph signals. In *2018 IEEE Statistical Signal Processing Workshop (SSP)*, pages 21–25. IEEE, 2018.

[30] Sunil K Narang, Akshay Gadde, and Antonio Ortega. Signal processing techniques for interpolation in graph structured data. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5445–5449. IEEE, 2013.

[31] Elvin Isufi and Geert Leus. Distributed sparsified graph filters for denoising and diffusion tasks. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 5865–5869. IEEE, 2017.

[32] Masaki Onuki, Shunsuke Ono, Masao Yamagishi, and Yuichi Tanaka. Graph signal denoising via trilateral filter on graph spectral domain. *IEEE Transactions on Signal and Information Processing over Networks*, 2(2):137–148, 2016.

[33] Fan Zhang and Edwin R Hancock. Graph spectral image smoothing using the heat kernel. *Pattern Recognition*, 41(11):3328–3342, 2008.

[34] Nicolas Tremblay, Gilles Puy, Rémi Gribonval, and Pierre Vandergheynst. Compressive spectral clustering. In *International Conference on Machine Learning*, pages 1002–1011, 2016.

[35] Ivana Tosic and Pascal Frossard. Dictionary learning. *IEEE Signal Processing Magazine*, 28(2):27–38, 2011.

[36] Stéphane G Mallat and Zhifeng Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on signal processing*, 41(12):3397–3415, 1993.

[37] Joel A Tropp. Greed is good: Algorithmic results for sparse approximation. *IEEE Transactions on Information theory*, 50(10):2231–2242, 2004.

[38] Scott Shaobing Chen, David L Donoho, and Michael A Saunders. Atomic decomposition by basis pursuit. *SIAM review*, 43(1):129–159, 2001.

[39] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological)*, 58(1):267–288, 1996.

[40] Irina F Gorodnitsky and Bhaskar D Rao. Sparse signal reconstruction from limited data using focuss: A re-weighted minimum norm algorithm. *IEEE Transactions on signal processing*, 45(3):600–616, 1997.

[41] David P Wipf and Bhaskar D Rao. Sparse bayesian learning for basis selection. *IEEE Transactions on Signal processing*, 52(8):2153–2164, 2004.

[42] Bruno A Olshausen and David J Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision research*, 37(23):3311–3325, 1997.

[43] Kjersti Engan, Sven Ole Aase, and J Hakon Husoy. Method of optimal directions for frame design. In *1999 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings. ICASSP99 (Cat. No. 99CH36258)*, volume 5, pages 2443–2446. IEEE, 1999.

[44] Michal Aharon, Michael Elad, and Alfred Bruckstein. K-svd: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on signal processing*, 54(11):4311–4322, 2006.

[45] Duc-Son Pham and Svetha Venkatesh. Joint learning and dictionary construction for pattern recognition. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[46] Qiang Zhang and Baoxin Li. Discriminative k-svd for dictionary learning in face recognition. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2691–2698. IEEE, 2010.

[47] Zhuolin Jiang, Zhe Lin, and Larry S Davis. Learning a discriminative dictionary for sparse coding via label consistent k-svd. In *CVPR 2011*, pages 1697–1704. IEEE, 2011.

[48] Julien Mairal, Francis Bach, Jean Ponce, Guillermo Sapiro, and Andrew Zisserman. Discriminative learned dictionaries for local image analysis. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2008.

[49] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):210–227, 2008.

[50] Xiao Ling, Wenyuan Dai, Gui-Rong Xue, Qiang Yang, and Yong Yu. Spectral domain-transfer learning. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 488–496, 2008.

[51] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[52] Min Xiao and Yuhong Guo. Feature space independent semi-supervised domain adaptation via kernel matching. *IEEE transactions on pattern analysis and machine intelligence*, 37(1):54–66, 2014.

[53] Huayan Wang and Qiang Yang. Transfer learning by structural analogy. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, 2011.

[54] Fan Zhu and Ling Shao. Weakly-supervised cross-domain dictionary learning for visual recognition. *International Journal of Computer Vision*, 109(1-2):42–59, 2014.

[55] Zhun Zhong, Zongmin Li, Runlin Li, and Xiaoxia Sun. Unsupervised domain adaptation dictionary learning for visual recognition. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 16–26. Springer, 2018.

[56] Songsong Wu, Xiao-Yuan Jing, Dong Yue, Jian Zhang, K Jian Yang, and Jingyu Yang. Unsupervised visual domain adaptation via dictionary evolution. In *2016 IEEE international conference on multimedia and expo (ICME)*, pages 1–6. IEEE, 2016.

[57] Yael Yankelevsky and Michael Elad. Dual graph regularized dictionary learning. *IEEE Transactions on Signal and Information Processing over Networks*, 2(4):611–624, 2016.

[58] David K Hammond, Pierre Vandergheynst, and Rémi Gribonval. Wavelets on graphs via spectral graph theory. *Applied and Computational Harmonic Analysis*, 30(2):129–150, 2011.

[59] Miao Zheng, Jiajun Bu, Chun Chen, Can Wang, Lijun Zhang, Guang Qiu, and Deng Cai. Graph regularized sparse coding for image representation. *IEEE transactions on image processing*, 20(5):1327–1336, 2010.

[60] Mingsheng Long, Guiguang Ding, Jianmin Wang, Jiaguang Sun, Yuchen Guo, and Philip S Yu. Transfer sparse coding for robust image representation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 407–414, 2013.

[61] Yael Yankelevsky and Michael Elad. Structure-aware classification using supervised dictionary learning. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4421–4425. IEEE, 2017.

[62] Yael Yankelevsky and Michael Elad. Graph-constrained supervised dictionary learning for multi-label classification. In *2016 IEEE International Conference on the Science of Electrical Engineering (ICSEE)*, pages 1–5. IEEE, 2016.

[63] Geoff Davis, Stephane Mallat, and Marco Avellaneda. Adaptive greedy approximations. *Constructive approximation*, 13(1):57–98, 1997.

[64] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[65] Jon W Tolle. Sequential quadratic programming. *Acta Numerica 1995: Volume 4*, 4:1–51, 1995.

[66] Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *Annals of statistics*, 32(2):407–499, 2004.