Master Project title

Final master project

Cristina Gava

July 19, 2018

Contents

Chapter 1

The emerging field of graph signal processing

Nowadays huge amounts of data are collected every day, from engineering sciences to our personal data regarding health monitoring, to financial data or political influences. The analysis and the process of these huge datasets has become a non indifferent challenge, since, in fact, data tends to reside on complex and irregular structures, which progressively required the introduction of innovative approaches to better handle and utilize them. [1] [2]

One of the central entities we will use in this work is the concept of *graph signal*. Graphs are generic data representation structures, which are useful in a great variety of fields and applications for everyday life and technology in general. These structures are composed by two main elements: the nodes and the edges; the former are a set of points, identifying an aspect of the graph structure itself, while the latter are connections between the nodes. Several structures we can encounter in natural entities and abstract constructions can be represented by a graph structure, and when we associate values to their set of nodes and edges we obtain a graph signal. To be specific, a graph signal is seen as a finite collection of samples located at each node in the structure and which are interconnected among themselves through the edges, to which we can associate numerical values representing the weights of the connections. The metric for these weights is not unique, as it depends on which relation between the nodes we are looking at: a typical metric for graph weights may be,

for example, inversely proportional to the distance (physical or not) between two different nodes, but other options are possible.

As previously mentioned, graph structures appear to be significantly helpful when they are used to represent signals and their applications are the most varied: we could focus on transportation networks, where we could want to work with data describing human migration patterns or trade goods; we could also apply graph theory over social networks, in which users can be seen, for example, as our nodes while their connections to friends are our edges. [1] Brain imaging is another interesting application of graph signals: through it we could infer the inner structure of some regions of the brain, in a way that permits us to better understand physiological dynamics. [3] We could go on listing a considerable amount of valuable applications, from genetic and biochemical research to fundamental physical experiments; what these big amounts of data have in common is the fact that their complex inner structure makes it difficult to apply most of the well-known tools, like Principal Component Analysis (PCA), spectral analysis or Singular Value Decomposition (SVD), without redefining the signal structure. [2]

1. Mathematical description of the graph structure

In the field of Graph Signal Processing (GSP), spectral graph theory plays an important role, it focuses on analysing, constructing and manipulating graphs and makes uses of well know tools and concepts as frequency spectrum and the graph Fourier transform. In our work, the signals we are considering are defined on an undirected, connected, weighted graph $\mathcal{G} = \{\mathcal{V}, \varepsilon, W\}$ consisting on a finite set of vertices $\mathcal{V}$ with $|\mathcal{V}| = N$, a set of edges ε and a weighted adjacency matrix $\mathcal{W}$. The adjacency matrix contains the information of the connections between two nodes: if an edge e is present between two vertices i and j, then the entry $W_{i,j}$ represents the weight of that edge, while if there is no connection between two nodes, the value of the edge is null ($W_{i,j} = 0$). Moreover, if the graph was not connected and had K connected components, then we could decompose the graph signal over $\mathcal{G}$ into M sections which can be processed each one independently of the other.

Over this structure, we can then lay **(Cri says: cerca un verbo migliore)** the signal we need to analyse: to be precise, a signal or function $f : \mathcal{V} \to \mathbb{R}$ defined on the vertices of our graph, can be

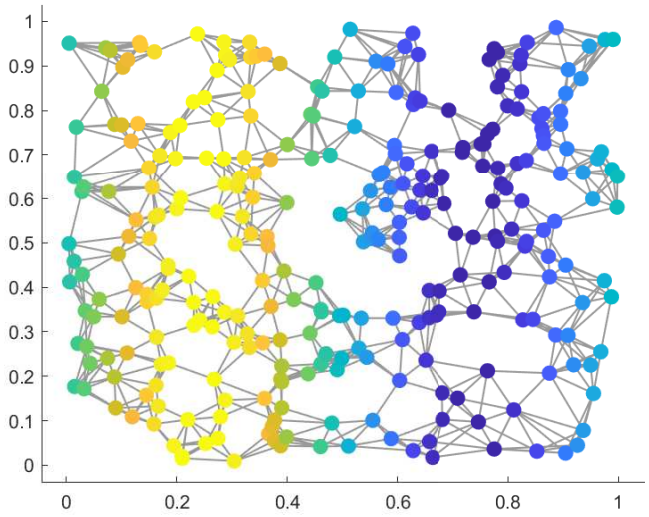


Figure 1.1: Random sensor graph

described as a vector $f \in \mathbb{R}^N$ that has the signal value at the i^{th} vertex in $\mathcal{V}$ as the i^{th} component of the vector f. Figure 1.1 shows an example of graph signal: the colours of the vertices represents the values the signal has on them.

2. the inverse covariance matrix

(Cri says: Vedere se aggiungere o meno questa sezione)

The graph Laplacian

To go from the graph vertex domain (which plays the same role of the time domain in the classical signal processing theory) to the spectral domain the *graph Laplacian operator*, or *Combinatorial graph Laplacian*, has been introduced, which is defined as $L := D - W$, where D represents a diagonal matrix whose i^{th} diagonal element is equal to the sum of weights of all the edges incident to the vertex i. This entity is a difference operator, since it satisfies the condition:

$$((Lf)(i) = \sum_{j \in \mathcal{N}_i} W_{i,j}[f(i) - f(j)] \tag{2.1}$$

The element $\mathcal{N}_i$ represents the set of vertices connected to vertex i by an edge.

From its definition, the graph Laplacian L is a real symmetric matrix, thus it has a complete set of orthonormal eigenvectors, here denoted by $\{u_l\}_{l=0,1,\dots,N-1}$, to which real and non-negative eigenvalues are associated ($\{\lambda_l\}_{l=0,1,\dots,N-1}$). Together with the eigenvectors, they satisfy the condition:

$$Lu_l = \lambda_l u_l, \text{ for } l = 0, 1, \dots, N-1 \tag{2.2}$$

In the eigenvalues set, the one corresponding to 0 has a multiplicity equal to the number of connected components of the graph; from this, since we are considering only connected graphs, we can assume the Laplacian eigenvalues have the distribution: $0 = \lambda_0 < \lambda_1 \leq \lambda_2 \cdots \leq \lambda_{N-1} = \lambda_{max}$ and, of course, the set $\sigma(L) = \{\lambda_0, \lambda_1, \dots, \lambda_{N-1}\}$ is the entire spectrum of our signal.

Eigenvalues and eigenvectors are then used to build up the graph version of a well known and useful transform, which is the *classical Fourier Transform* and that we recall here being defined as:

$$\hat{f}(\xi) = \langle f, e^{2\pi i \xi t} \rangle = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt \tag{2.3}$$

and that we can see as the expansion of a function f in terms of the complex exponentials, elements which represent the eigenfunctions of the one-dimensional Laplace operator:

$$-\Delta(e^{2\pi i \xi t}) = -\frac{\partial^2}{\partial t^2} e^{2\pi i \xi t} = (2\pi\xi)^2 e^{2\pi i \xi t} \tag{2.4}$$

From the observation of the classical Fourier Transform, we can analogously define the *Graph Fourier Transform* $\hat{f}$ of any function $f \in \mathbb{R}^N$ on the vertices of $\mathcal{G}$ as the expansion of f in terms of the eigenvectors of the graph Laplacian [3]:

$$\hat{f}(\lambda_l) = \langle f, u_l \rangle = \sum_{i=1}^{N} f(i) u_l * (i) \tag{2.5}$$

while, at the same time, the *inverse Graph Fourier Transform* is given by:

$$f(i) = \sum_{l=0}^{N-1} \hat{f}(\lambda_l) u_l(i) \tag{2.6}$$

Chapter 2

The Graph learning issue

For the most part, the research regarding graph signal processing has been focusing on working with the signals spanned onto a graph manifold, assuming the graph structure to be already known. However, the choice of the underlying graph for the signal not necessarily represents faithfully the ground truth intrinsic manifold over which the signal is structured. In these cases, a more suitable approach to the problem could be trying to learn the graph structure underneath, such that the processing of the signal can rely on assumptions that better capture the intrinsic relationships between the entities or make them clear when otherwise it would not have be so. [4] Clearly the question is not trivial, since in general the action of learning a graph from sample data represents an ill-posed problem, with the consequence that there may be many solutions to the structure associated to a data. [5] To overcome this obstacle several approaches have been designed in the recent years, and to some of them we will give a more detailed description in the next chapters, when we will address the basis of our work.

Representing a graph: graph learning techniques and dictionary representation

In the previous section we presented the main reason why signal processing field expressed the necessity of having new different structures in order to better represent the amount of information we can collect from a phenomena: what we did not focused on yet, is the fact that these amount of data are usually largely redundant, since they are a densely sampled version of the signal and they may represent multiple correlated versions of the same physical event. So normally the significative information regarding the underlying processes is largely reducible in dimensionality with respect of the collected dataset. [6] Thus, we can obtain the data representations starting from the idea that our observations can be described by a sparse subset of elementary signals - so called *atoms* - taken from an *overcomplete dictionary*. When the dictionary forms a basis, then every signal can be univocally represented through the linear combination of the dictionary atoms, moreover the overcompleteness property **(Cri says: definisci i significati delle n. Sei sicura che stai avendo a che fare con dizionari overcomplete? Si si)** implies that atoms are linearly dependent. [6] [7]

With these premises, we consider a dictionary $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \dots, \mathbf{d}_L] \in \mathbb{R}^{N \times L}$, in which the columns represent the dictionary atoms and, for the overcompleteness, $L \geq N$. Through this entity, the signal representations can be done through two main paths, either the *synthesis* path, or the *analysis* path, and the two can significantly differ in the overcomplete case.

In the synthesis path, the signal $\mathbf{x} \in \mathbb{R}^N$ is represented as a linear combination of the dictionary atoms:

$$\mathbf{x} = \mathbf{D}^T \gamma_s \tag{0.1}$$

while in the analysis path it is represented through its inner product with the atoms:

$$\gamma_a = \mathbf{D}^T \mathbf{x} \tag{0.2}$$

The representation in 0.1 has the consequence that, when the dictionary is overcomplete, the set of representations $\gamma_s$satisfying the equation is *infinitely large*, allowing us to look for the most informative representation of the signal with respect of a certain cost function $\mathcal{C}(\gamma)$. In this way we arrive to a first general optimization problem in the form:

$$\gamma_s = \underset{\gamma}{\operatorname{argmin}}\ \mathcal{C}(\gamma) \quad \text{Subject To } \mathbf{x} = D\gamma \tag{0.3}$$

The way we choose the form of the cost function obviously influences the structure of our solution, to be specific, one of our goals is to achieve sparsity in the representation of the signal, such that the signal reconstruction is reduced in dimensionality as we are trying to achieve from the beginning. Problem in 0.3 becomes what is commonly referred as *sparse coding*, and there are different functions we can apply in order to obtain it: these functions have the characteristic of being tolerant to large coefficients and at the same time importantly penalize small non-zero ones. Among these functions, our choice fell on the l norm, which is one of the simplest and most effective function of this type.

1. Choosing the right dictionary

What we did not focused on yet is the choice of the proper Dictionary for out task. In the research there has been so far, different models of Dictionaries have been defined and used for the most different purposes, in the beginning the attention was mainly on traditional dictionaries, such as wavelet and Fourier dictionaries, which are simple to use and perform well for 1-dimensional signals. However, these structures were too simple to properly describe more complex and high-dimensional data, so the focus slowly moved to seeking solutions that better performed in this environment. Dictionaries emerged from this need were coming from two main sources:

- As an *analytical model*, which allows to extract the dictionary straight form the data for a fast implicit implementation that does not require multiplication by the dictionary matrix;
- As a *set of realizations* of the data, which offers an increased flexibility and adaptability to data structure;

The first model prefers speed over adaptability, since its success depends on the chosen underlying model, sometimes resulting in a over-simplistic solution for the proposed problem. From this, the necessity of also focusing on the second type of dictionary, also defined as *trained dictionary*.
Machine learning techniques of the period between 1980's and 1990's allowed to approach this problem under the new assumption that the structure of a natural phenomena can be accurately extracted *straight form the data* with better results than using a mathematical formulation. The most recent training methods focus on the l^0 and l^1 sparsity measures, which have a simpler formulation and at the same time can use the more recent sparsity coding techniques. [8] [9] Among these learning methods, particular relevance acquired the *parametric training methods*, such as *translation-invariant dictionaries*, *multiscale dictionaries* or *sparse dictionaries*. These implementations involve the reduction of parameters' number and the assumption of several desirable properties on the dictionary, in the end leading to an accelerate convergence, reduced density of the local minima and the convergence to a better solution. Moreover, the generalization of the learning process is improved thanks to the smaller number of parameters, as much as the reduction in number of examples needed. Finally, parametric dictionaries bring to a more efficient implementation, due to the fact that parametrization typically has a more compact representation and, not less important, a parametric dictionary may be designed to represent infinite or arbitrary-sized signals. [7] **(Cri says: forse aggiungi parte sugli sparse dictionaries sempre presa dallo stesso paper. Poi: vedi se aggiungere un excursus storico sui dizionari ad apprendimento)**.

The sparse approximation

The intention of sparse approximations to represent a certain signal y of dimension n as the linear combination of a small amount of signals selected from the source database which is the dictionary and in which the elements typically are unit norm functions called *atoms* and can be denoted through ϕ_k with $k = 1, \ldots, N$ and being N the size of the dictionary. When a dictionary is overcomplete, then every signal can be represented with the aforementioned linear combination:

$$\mathbf{y} = \phi\mathbf{a} = \sum_{k=1}^{N} a_k \phi_k \tag{1.4}$$

and the value that $\mathbf{a}$ can assume is not unique.

To achieve sparse and efficient representations, the requirement for finding the exact representation is in general relaxed: starting from the NP-hard problem which is the minimization of the l^0 norm of $\mathbf{a}$, we can approach the issue through convex relaxation methods that solve a problem in the form:

$$\min_{\mathbf{a}} \ (||\mathbf{y} - \phi\mathbf{a}||_2^2 + \lambda||\mathbf{a}||_1) \tag{1.5}$$

in which the relaxation allowed to replace the nonconvex l^0 norm in the original problem with a more meaningful l^1 norm. [6]

2. The correspondence between the dictionary and the graph signal

In 2. we introduced the Laplacian operator as $L = D - W$, here we focus on a modified version of this operator, that is the *normalized Laplacian* $\mathcal{L} = D^{-\frac{1}{2}}LD^{-\frac{1}{2}}$, also a real symmetric and positive semidefinite matrix, with a complete set of orthonormal eigenvectors $\chi = [\chi_0, \chi_1, \ldots, \chi_{N-1}]$ and associated non negative eigenvalues $\sigma(\mathcal{L}) := \{0 = \lambda_0, \lambda_1, \ldots, \lambda N - 1 \leq 2\}$ (where the upper bound 2 comes from the normalization). The normalized Laplacian eigenvectors are Fourier basis, this bringing any function y defined on the vertices of the graph to be represented in his Fourier transform version $\hat{y}$ at frequency λ_l as:

$$\hat{y}(\lambda_l) = \langle y, \chi_l \rangle = \sum_{n=1}^{N} y(n)\chi_l^*(n) \tag{2.6}$$

and its relative inverse transform to be:

$$y(n) = \sum_{l=0}^{N-1} \hat{y}(\lambda_l)\chi_l(n), \quad \forall n \in \mathcal{V} \tag{2.7}$$

This transform plays a major role not only in the harmonic analysis, but also in defining the signal translation onto a graph. It is worth to notice here that in the classical signal processing field the translation operator is defined through the change of variable $(T_n f)(t) := f(t - n)$, but when it comes to graph signal processing, this concept looses meaning since there is no significate to $f(\circ - n)$ in the graph environment. However, from transform theory we can recall that the classical translation operator T_n can also be seen as a convolution with a Kronecker delta δ centered in n[3]

[10]:

$$T_n g = \sqrt{N}(g * \delta_n) = \sqrt{N} \sum_{l=0}^{N-1} \hat{g}(\lambda_l)\chi_l^*(n)\chi_l \tag{2.8}$$

this leading to think about 2.8 as an operator acting on the kernel $g(\cdot)$ directly defined in the spectral domain and from which we can obtain the effective translation to kernel n through inverse graph Fourier transform. Moreover the term $\sqrt{N}$ in the Equation 2.8 is a guarantee for the translation operator to preserve the mean of the signal, while the $g(\cdot)$ kernel smoothness is a measure to control the localization of $T_n g$ centered at vertex n (meaning that the magnitude $(T_n g)(\mathrm{i})$ of the translated kernel at vertex i decays with the increasing distance from the center of the kernel). We can thus compose our atoms $T_n g$ as smooth entities, coming from the assumption that the kernel function in 2.8 is a smooth polynomial function of degree K:

$$\hat{g}(\lambda_l) = \sum_{k=0}^{K} \alpha_k \lambda_l^k, \quad l = 0, \ldots, N-1 \tag{2.9}$$

From this fact, we can so obtain a global expression for a translation operator as:

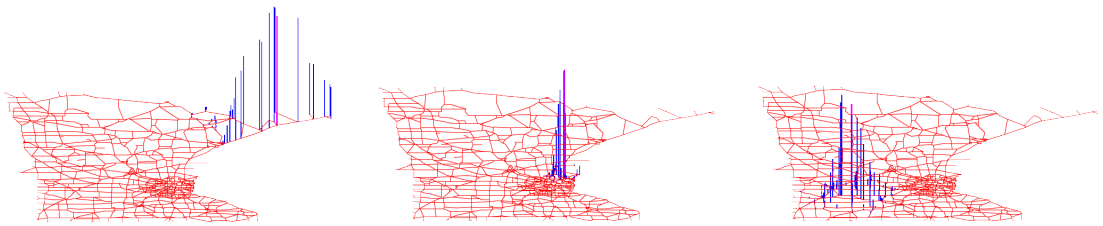
$$T_n g = \sqrt{N}(g * \delta_n) = \sqrt{N} \sum_{l=0}^{N-1} \sum_{k=0}^{K} \alpha_k \lambda_l^k \chi_l^*(n)\chi_l \tag{2.10}$$

$$= \sqrt{N} \sum_{k=0}^{K} \alpha_k \sum_{l=0}^{N-1} \lambda_l^k \chi_l^*(n)\chi_l = \sqrt{N} \sum_{k=0}^{K} \alpha_k (\mathcal{L}^k)_n \tag{2.11}$$

where $(\mathcal{L})_n$ represents the n^{th} column of the k^{th} power of the Laplacian matrix $\mathcal{L}^k$. The final concatenation of N such columns brings us to generate a set of N atoms, corresponding to the columns of

$$Tg = \sqrt{N}\hat{g}(\mathcal{L}) = \sqrt{N}\chi\hat{g}(\Lambda)\chi^T = \sqrt{N} \sum_{k=0}^{K} \alpha_k \mathcal{L}^k \tag{2.12}$$

Where the quantity Λ corresponds to the diagonal matrix of the eigenvalues in such a way that the following relation holds: $\mathcal{L} = \chi\Lambda\chi^T$ [5]. This brief section has the intrinsic meaning that if the kernel $g(\cdot)$ is a polynomial of degree K, then the translation operator is 0 in all the vertices i that are more than K hops far away from the center vertex n, which means that the vertex domain support of the translated vertex is contained in a sphere of radius K and center in n and its magnitude decades with the increasing distance from n to i. In Figure 2.1 there is an example of an atom translation in three different ways.



(a) (b) (c)

Figure 2.1: Example of graph signal translation: translation for $T_{200}g$ (a), $T_{1000}g$ (b) and $T_{2000}g$ (c)

Chapter 3

The smoothness assumption

Going further with the spectral analysis, some other intuitions about the graph spectrum can be conveyed from the from the classical setting to the graph setting.

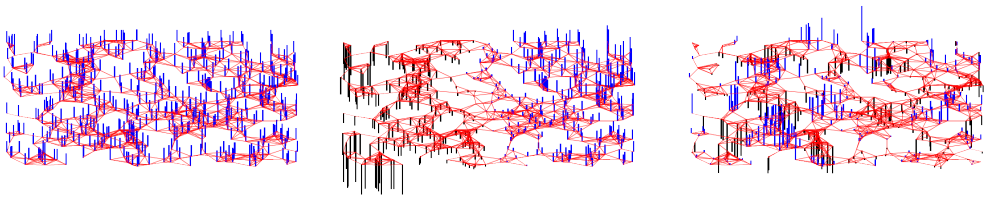
One of them is that in the classical setting the eigenvalues carry a specific notion of frequency: complex exponentials related to lower frequencies are relatively smooth, slowly oscillating functions whereas for high frequencies the associated complex exponentials oscillate more rapidly. Reporting this onto the graph spectral settings, we see that the Laplacian eigenfunctions associated with lower eigenvalues can be seen as smooth, while the ones related to higher eigenvalues tend to oscillate more intensively. [11] In the specific case of the null eigenvalue the related eigenvector (χ_0) is constant and equal to $\frac{1}{\sqrt{N}}$ [3].

Taking a step back, one of the common simple assumptions about data residing on graphs, is that the signal changes smoothly between connected nodes; in this, a simple way to quantify how smooth is a set of vectors residing on a weighted undirected graph, is through the function:

$$\frac{1}{2}\sum_{i,j} W_{i,j}||\chi_i - \chi_j||^2 \tag{0.1}$$

with W_ij representing the weight between node i and node j, as usual, and χ_i and χ_j representing two vectors in $\chi_1, \chi_2, \ldots, \chi_N \in \mathbb{R}$. This means that if two vectors χ_i and χ_j reside on two nodes connected by a high W_{ij}, then they are expected to have a small distance. Equation 0.1 can be thus rewritten in matrix form as [12] :

$$\mathrm{tr}(\chi^T L \chi) \tag{0.2}$$



$\mathbf{u}_0$ $\mathbf{u}_1$ $\mathbf{u}_{50}$

Figure 3.1: three graph Laplacian eigenvectors for a random sensor network graph. It can be seen how χ_{50} contains definitely more zero crossing than the other two vectors χ_0 and χ_1

and, recalling that the Laplacian matrix can be expressed as:

$$L = \chi\Lambda\chi \tag{0.3}$$

with Λ being the diagonal matrix containing the associated eigenvalues of the Laplacian, then we can insert 0.3 in 0.2, obtaining:

$$\text{tr}(\chi^T L\chi) = \text{tr}(\chi^T\chi\Lambda\chi^T\chi) = \text{tr}(\Lambda) \tag{0.4}$$

This result clearly explains the relation between the signal smoothness and the eigenvalues of the Laplacian that we presented at the beginning of the paragraph. Figure 3.1 shows different graph Laplacian eigenvectors for a random graph. The eigenvectors associated to larger eigenvalues tend to oscillate more rapidly and to have dissimilar values on vertices connected by a high weight edge, this moreover bringing to higher occurrences of zero crossing associated to the graph Laplacian. [3].

1. Smoothness and sparsity

In [12] the previous notion of smoothness is directly connected with the one of graph sparsity: in fact they show that the smoothness term can be seen equivalently as a weighted l^1 norm of the adjacency matrix, which being minimized can lead to a graph with a sparse set of edges, that prefers only the ones associated to small distances in the *pairwise distances matrix* $\mathbb{Z} \in \mathbb{R}_+^{N\times N}$ defined as $\mathbb{Z}_{i,j} = ||\chi_i - \chi_j||^2$.

This concept is strictly connected with the compressibility property of a signal, since previous studies validate the hypothesis that smooth signals are likely to be compressible in the frequency

domain. In [13] it has been shown that the liner approximation error is upper bounded by the overall variation of the signal, this bringing to the conclusion that if the total variation of a signal is small, then the approximation error is small. This resulting in the fact that the signal can be well approximated by a small portion of the Fourier coefficients.

(Cri says: vedi dopo se scrivere qualcosa)

Chapter 4

The algorithms we started from: Dorina's and Hermina's

With the

Chapter 5

Problem Presentation

Given the premises presented in the previous chapters, our work mainly focused on two open issues regarding both the dictionary learning and the graph learning problem. The starting point were the works of Dorina Thanou- regarding the dictionary learning approach for graph signals - ant the work of Hermina Petric Maretic - regarding the graph learning approach.
In the related work, Thanou et al. proposed a learning algorithm which was able to retrieve the kernel functions for a certain dictionary, alternately optimizing over the kernel coefficients α and the sparsity matrix X, while Maretic et al. started from the complementary assumption that the entities know in their case where exactly the kernel coefficients, while they alternatively optimized over the sparsity matrix and the graph weight matrix.

1. State of the art

There has already been a discrete amount of work around graph inference for signals that are expected to be smooth on the graph. In [5] Dong et al. addressed the problem adopting a factor analysis model for graph signals and imposing a Gaussian probabilistic prior on the independent variables which model the signal. Their results showed that smoothness property of a graph signal is favoured by the efficient representation obtained through these priors. In particular, in the algorithm they proposed they deployed the use of l^1 and Frobenius norm, the former is expressed by a constraint on the Trace of the Laplacian, and eventually accounts for a sparsity term, while the

latter is added as a penalty term in the objective function in order to control the distribution of the off-diagonal entries in L.

At the same time, in [12] Kalofolias proposes another framework to learn a graph structure under the smoothness assumption. They used as well the minimization of the trace term for the Laplacian matrix, clinching that the minimizations of it brings to naturally sparse solutions.

However, this assumption is not always exhaustively descriptive of the real problem we are considering, so we might want to add other reinforcements to the priors in order to better describe the situation. For example, we could imagine a real world network to show a localized behavior or a more organized one having a certain pattern repeating over the graph. In [4], this was an assumption for the signal, bringing it to be a sparse combination of a small number of atoms in a polynomial graph dictionary. This in the end was creating atoms localized in every node on the graph, repeating the same pattern over it with respect to every node as a source, in the same way it is described in [10]. The limitation Thanou's work is mainly the fact that, as we already mentioned, it assumes the kernels to be know function while it concentrates only on the graph dictionary and thus fixing the spectral behaviour of the atoms.

2. Problem description

As previously mentioned, this work deals with the general class of signals that can be represented in a sparse way through atoms of a polynomial dictionary [10], such that it can give a natural segregation of each signal into localised components, as shown in Figure 5.1. The figure shows how each of these components derive from one atom directly dependent of the graph dictionary and this is well explained if, for polynomial kernels of degree k, we see our atoms as an entities spanning only the k-hop distance from the node representing the center of the atom (source).

2.1 Assumptions

(Cri says: Verifica la faccenda della community structure che stai per scrivere) In opposition to the prior work, here we assume the sources of our atoms to be known and the same for all signals, meaning that the network has a certain amount of nodes acting like sources an thus controlling the behavior of the network. Another strictly related assumption, is that the graph has a strong

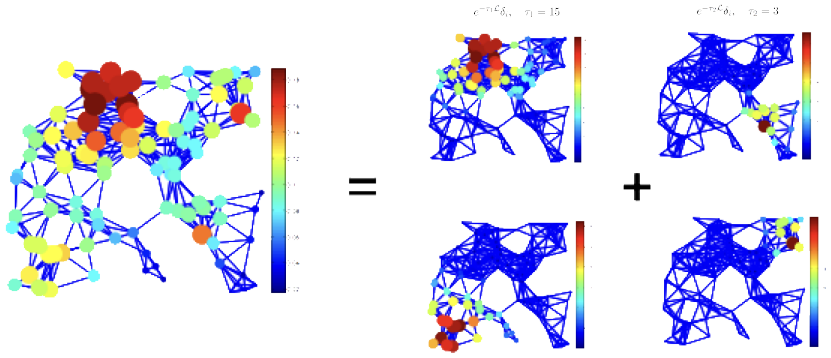


Figure 5.1: Example of signal decomposition using the polynomial dictionary

community structure, meaning that the graph can be separated into communities which are poorly related among themselves, but hold strong connections inside them. These two assumptions are somewhat exchangeable and the reduction of one of them could be a future interesting aspect to better develop.

2.2 Representation graph

2.3 Dictionary learning section

If we considered the former formulation for our problem, we would ignore the fact that atoms in D_{small} actually are coming from a graph dictionary, while this is on the contrary something we would like to include in our problem. In fact, we did assume these atoms to be vectors describing the signal, but we kept this part separated from the graph notion. We start from stating that we want that our atoms are representative of the portion of the graph they are covering, and so they should spread with non trivial values over all nodes of the subgraph. This reason mad us constraint our dictionary kernels in a way that they have mostly smooth support in the surroundings of the source node. This information is something different form the smoothness assumption we widely examined in the previous sections, since it adds some information strictly related to the behavior of the kernel itself. In fact, if we learned a smaller order polynomial with constraints in high frequencies, we would not have a large spread of our atoms, things we would like to have since in this way our atoms

can represent a large portion of the graph. **(Cri says: rivedila ti prego)**

To have this new smoothness constraint, we look at the kernel polynomial:

$$g(\lambda) = \sum_{k=0}^{K} \alpha_k \lambda_k \tag{2.1}$$

and we try to constraint the roots to correspond to the highest eigenvalues. This would change the expression of the kernel and turn our learning problem into a simpler one, since now we are trying to learn a smaller number of coefficients:

$$g(\lambda) = h(\lambda)(\lambda - \lambda_n)(\lambda - \lambda_{n-1}) \cdot \cdots \cdot (\lambda - \lambda n - l + 1) \tag{2.2}$$

$$where \qquad h(\lambda) = \sum_{k=0} K - l\gamma_k \lambda^k \tag{2.3}$$

This formula them can bring to a dictionary problem of this type:

$$\arg \min_{\gamma_0,\ldots,\gamma_{K-l} X_{small}} ||Y - D_{small} X_{small}||_F^2 + \alpha ||D_{small}||_1 \tag{2.4}$$

$$\begin{aligned} with \qquad g(\lambda) &= h(\lambda)(\lambda - \lambda_n)(\lambda - \lambda_{n-1}) \cdot \cdots \cdot (\lambda - \lambda n - l + 1) \\ h(\lambda) &= \sum_{k=0} K - l\gamma_k \lambda^k \\ D_i &= [g(L_i)]_{\text{source}_i} \\ 0 \preceq g(\lambda) &\preceq c \qquad (2.5) \\ (c - \epsilon)I \preceq \sum_{s=1}^{S} D_s &\preceq (c + \epsilon)I \qquad (2.6) \end{aligned}$$

Where the positions of the non-trivial values of X_{small} are known, but the values are to be learned, while the second term in Equation 2.4 accounts for the graph sparsity. Moreover, the last two constraints 2.5 and 2.6 are taken from [10] and ensure the kernels are bounded and span the entire spectrum.

2.4 Joint large graph and dictionary learning

So came to this point, an interesting aspect we could focus on was the attempt to joint both the graph and the dictionary learning problem; in fact, if we assume that the graph is unknown we

could imagine adding a graph learning part to the optimization. In doing this, we had to take into account two main complications: first the fact that we didn't have fixed eigenvalues, since we would have learned the Laplacian again at every new optimization step, and second the fact that as we are learning the eigenvalues, we don't have their initial values either. To this aspects we still have to take into account that the structure of the kernels should be incorporated in our learning problem. For this part, in the end we assumed only one kernel for simplicity so that the dictionary atoms will have all the same kernel but will be localized in different sources spreading throughout different parts of the graph. This meaning that the pattern has to be followed by all the atoms, but it will be adapted to small different graphs.

Therefore the overall problem could become something similar to:

$$\arg \min_{X, W_1, \dots, W_m, \gamma_0, \dots, \gamma_{k-l}} ||Y - DX||_F^2 + \alpha ||D||_1 \tag{2.7}$$

where
$$\begin{aligned} g(\lambda) &= h(\lambda)(\lambda - \lambda_n)(\lambda - \lambda_{n-1}) \cdot \dots \cdot (\lambda - \lambda n - l + 1) \\ h(\lambda) &= \sum_{k=0} K - l\gamma_k \lambda^k \\ D_i &= [g(L_i)]_{\text{source}_i} \\ L_i &= \text{normalised Laplacian}(W_i) \\ W_{ij} &= W_{ji} \geq 0, \quad \forall i, j \qquad (2.8) \\ W_{ii} &= 0, \quad \forall i \\ 0 \leq g(\lambda) &\leq c \qquad (2.9) \\ (c - \epsilon)I \preceq \sum_{s=1}^{S} D_s &\preceq (c + \epsilon)I \qquad (2.10) \end{aligned}$$

Where we remember the constraint in 2.8 accounts for the fact that we are assuming to work with undirected graphs.

2.5 Work structure

To summarize, the main steps in which the following work is articulated are the following:

- We implemented some constraints related to the smoothness of the kernels in the dictionary learning approach, and we showed the improvements deriving from it;
- We presented a solution to solve both the dictionary and the graph learning problem within the same framework;
- We underlined how the smoothness constraints on the kernels can still provide improvements even in the case of double learning frameworks;
- We briefly present the possible next steps in this work, in order to obtain further implementations;

Chapter 6

The dictionary learning improvement under smoothness constraints

1. The point we started from: "Learning Parametric Dictionaries for Signals of Graphs"

As we previously hinted, in the work of Thanou, Shuman and Frossard the focus was on learning the parametric functions (*kernels*) through which it was possible to reconstruct the graph signal. Moreover, in their work they modelled the graph signal as a combination of overlapping local patterns, describing localized events on the graph which can appear in several vertices. This representation of the graph signal brought to a representation of the dictionary structure that as a concatenation of subdictionaries that are polynomials of the graph Laplacian. Under this assumption, they then learned the coefficients of the kernels through a numerical optimization.

1.1 The dictionary structure and main assumptions

To be specific, the dictionary structure we used which is presented in [10] is in the following form.: we designed a graph dictionary $\mathcal{D} = [\mathcal{D}_1, \mathcal{D}_2, \ldots, \mathcal{D}_S]$, or a concatenation of a set of S subdictionaries, as:

$$\mathcal{D}_s = \hat{g}_s(\mathcal{L}) = \chi(\sum_{k=0}^{K} \alpha_{sk}\Lambda^k)\chi^T = \sum_{k=0}^{K} \alpha_{sk}\mathcal{L}^k \tag{1.1}$$

where $\hat{g_s}$ is the pattern (*the kernel*) of the subdictionary $\mathcal{D}_s$ and where the atom given by the n^{th} column of subdictionary $\mathcal{D}_s$ is equal to $\frac{1}{\sqrt{N}}T_n g_s$, the translation operator defined in Equation 2.8 divided by the constant $\frac{1}{\sqrt{N}}$.
However, despite the polynomial constraint guarantees the localization of the atoms in the vertex domain, it does not provide though any information about the spectral representation of the atoms. For this reason we included in our problem also the two matrix inequalities already listed in the constraints list of the optimization problem in 2.7:

$$0 \preceq \mathcal{D}_s \preceq cI, \quad \forall s \in \{1, 2, \dots, S\}, \tag{1.2}$$

and (1.3)

$$(c-\epsilon)I \preceq \sum_{s=1}^{S} D_s \preceq (c+\epsilon)I \tag{1.4}$$

Equation 1.2 accounts for the fact that our kernels are nonnegative and uniformly bounded by a given constant c (which we will further suppose equal to 1), while Equation 1.4 accounts for the fact that the signals we consider usually contain frequency components spread across the entire spectrum and so the kernels should cover it as well. These two spectral constraints together contribute to increase the stability of the dictionary.

Finally, we underline the fact that for the design of the dictionary the eigenvectors used are the ones from the normalized graph Laplacian, since this configuration revealed to be more suitable for our bodywork.

2. The dictionary learning algorithm under smoothness constraints

With these basis we therefore implemented a dictionary learning algorithm based on the one presented in [10] in order to add the smoothness priors. As previously mentioned, we can cast the

general problem as the following optimization problem:

$$\begin{aligned} \arg \min_{\alpha \in \mathbb{R}^{(K+1)*S},\, X \in \mathbb{R}^{SN \times N}} & ||Y - DX||_F^2 + \mu ||\alpha||_2^2 \\ \text{subject to} \quad & ||x_m||_0 = T_0, \quad \forall m \in \{1, \ldots, M\} \\ & \mathcal{D}_s = \sum_{k=0}^{K} \alpha_{sk}{}^{\|}, \quad \forall s \in \{1, 2, \ldots, S\} \\ & 0 \preceq \mathcal{D}_s \preceq cI, \quad \forall s \in \{1, 2, \ldots, S\}, \\ & (c - \epsilon)I \preceq \sum_{s=1}^{S} D_s \preceq (c + \epsilon)I \end{aligned} \tag{2.5}$$

with μ being the coherence coefficient **(Cri says: verifica se è la coherence ciò di cui si parla qui)** x_m corresponding to the m^{th} column of the sparsity matrix X, T_0 being the sparsity level of the coefficient of each signal and the l_0 norm term one time more standing for the sparsity need we manifested before.

Since the optimization problem is non convex, there is the need to approximate it by the alternation of two optimization convex subproblems, namely the sparse coding and the dictionary update step.

The sparse coding step aims at learning the sparsity matrix X solving

$$\underset{X}{\text{argmin}} \, ||Y - DX||_F^2 \qquad \text{subject to} \quad ||x_m||_0 \leq T_0, \tag{2.6}$$

and it is solved using the Orthogonal Matching Pursuit (OMP) method, an iterative greedy algorithm that at each step selects the element in the dictionary that is best correlated with the signal residual part; it subsequently produces a new estimation through the projection of the signal onto the dictionary elements that have already been selected. This method admits simple and fast implementation and it has been proved that his error on the approximation is only a small factor worse than the minimal error obtained through a non-sparse representation. [14].

On the other side, the dictionary update step aims at finding the polynomial coefficients of the

kernels by minimizing

$$\arg \min_{\alpha \in \mathbb{R}^{(K+1)S}} ||Y - DX||_F^2 + \mu||\alpha||_2^2 \tag{2.7}$$

subject to the remaining constraints in Equation 2.5 **(Cri says: in caso più avanti aggiungi la parte su P_{nm}^s se vedi che si rivela utile)**

To this main problem we then added the smoothness prior through two different implementations, the first one trying to insert the smoothness assumption inside the objective function of the dictionary update step, while the second one focusing on adding a smoothness prior in the constraints of the optimization function.

2.1 Smoothness prior in the objective function

We remember the equations 2.3, and we reformulate them as

$$g(\lambda) = \sum_{k=0}^{K} \alpha_k \lambda^k = \sum_{n=0}^{N} \gamma_n \lambda^n \times \sum_{m=0}^{M} \beta_m \lambda^m \tag{2.8}$$

With $h(\lambda) = \sum_{n=n}^{N} \gamma_n \lambda^n$ and $(\lambda - \lambda_M)(\lambda - \lambda_{M-1}) \cdot ... \cdot (\lambda - \lambda_{M-N+1})$ being the two subpolynomials in which 2.8 has been decomposed. B Between these, $h(\lambda)$ represents the polynomial with the unknown coefficients to be found, while the polynomial in the unknown β is the one obtained from imposing the highest eigenvalues as roots of the general polynomial 2.8 **(Cri says: torna nelle sezioni indietro per sistemare gli indici del polinomio h)** Since we deal with the optimization problem in Equation 2.5 through linear matrix inequality tools, we try to model the vector containing the kernel coefficients in such a way that its structure itself already holds the information regarding the structure of the two sub polynomials. To do so, we multiply the two sub polynomials among each other without resolving the equation with known β, but leaving them as unknown, in such a way that the behavior of the overall polynomial emerges. We thus obtain the structure that every α

coefficient should have in order to transmit the smoothness prior to the objective function. Namely:

$$\begin{aligned}\sum_{n=0}^{N}\gamma_n\lambda^n \times \sum_{m=0}^{M}\beta_m\lambda^m = (\beta_0\gamma_0)\lambda^0 + (\beta_0\gamma_1+\beta_1\gamma_0)\lambda + \cdots + \\ (\beta_0\gamma_N+\beta_1\gamma_{N-1}+\cdots+\beta_N\gamma_0)\lambda^N + \cdots + \\ (\beta_1\gamma_N+\beta_2\gamma_{N-1}+\cdots+\beta_{N+1}\gamma_0)\lambda^{N+1} + \cdots + \\ (\beta_N\gamma_N+\beta_{N+1}\gamma_{N-1}+\cdots+\beta_M\gamma_0)\lambda^M + \\ (\beta_{N+1}\gamma_N+\cdots+\beta_{2N+1}\gamma_1)\lambda_{M+1} + \cdots + \\ (\beta_M\gamma_N)\lambda^K\end{aligned} \tag{2.9}$$

Once we obtained it, we insert this knew knowledge over the kernels structure in the optimization objective function, where now the alpha vector has a reduced number of unknowns that have to be found. With the problem set in this way we can then analyse the new results we obtain. **(Cri says: verifica se nei punti precendenti hai dato abbastanza spazio a questo concetto di smoothness e di come si comporta un grapho soggetto a questa struttura, se non è così ribidisci il fatto che ci interessa sapere altro sulla smth perchè così il grafo risulta più frammentato e in passi successivi separarlo ptrebbe essere un notevole vantaggio)**

2.2 Smoothness prior in the optimization constraints

(Cri says: qui spiega meglio la motivazione quando avrai una funzione decente per l'altra soluzione) Another approach to the problem is to insert this smoothness prior in the constraints of the optimization function, on one side this choice should be discarded in favour of the first approach, since adding other constraints to our already ill posed problem could add more non-desired complexity; on another side this choice could reveal motivated by the fact that the control over the behavior of the optimization algorithm is in this way more verifiable than in the previous case, in which a black-box-stile coefficients vector detains all the smoothness priors. In this case the constraint added is the imposition that the Vandermonde matrix, obtained selecting only the last M high-frequency eigenvalues in the graph Laplacian, should of course give a null vector when multiplied by the vector

of the alpha coefficients. If P is the total number of nodes in the considered graph, then we have:

$$\begin{bmatrix} 1 & \lambda_{P-M} & \lambda_{P-M}^2 & \dots & \lambda_{P-M}^K \\ 1 & \lambda_{P-M+1} & \lambda_{P-M+1}^2 & \dots & \lambda_{P-M+1}^K \\ \vdots & & & & \\ 1 & \lambda_P & \lambda_P^2 & \dots & \lambda_P^K \end{bmatrix} \begin{bmatrix} \alpha_0 \\ \alpha_1 \\ \vdots \\ \alpha_K \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{2.10}$$

2.3 The optimization algorithm

The final optimization algorithm is in this way described: **(Cri says: sistema la faccenda della caption in questo algoritmo)**

procedure INITIALIZATION
    $Y \leftarrow$ Signal samples set
    $T_0 \leftarrow$ Target sparsity
    $K \leftarrow$ Polynomial degree
    $S \leftarrow$ Number of subdictionaries
    $iterNum \leftarrow$ Number of iterations
    $\mathcal{D} \leftarrow$ Initial dictionary computed thirugh random kernels
end procedure
for $i = 1, 2, \ldots, iter$ **do**
    procedure SPARSITY STEP
        Scale each atom in $\mathcal{D}$ to a unit norm
        $X \leftarrow$ Sparsity estimation with OMP
        Rescale X and $\mathcal{D}$ to recover the polynomial structure
    end procedure
    procedure DICTIONARY LEARNING STEP $\alpha \leftarrow$ kernels estimation through sdpt3
    end procedure
    procedure DICTIONARY UPDATE STEP
        $\mathcal{D} \leftarrow \sum_{k=0}^{K} \alpha_k \mathcal{L}^k$
    end procedure
end for

2.4 Results

Data Generation

The datasets used in the experiments are composed by a signal Y, an adjacency matrix W and the comparison kernels used after the algorithm execution to verify the quality of the results. The signal Y is obtained as the repetition of 2000 samples randomly generated by a uniform distribution, the graph manifold onto which the signal is structured has 30 nodes (for simplicity reasons) while the

edges that are elements of the weight matrix W are generated from the gaussian function

$$w_{ij} = e^{-\frac{||x_i - x_j||_2^2}{2\sigma^2}} \tag{2.11}$$

following the approach in [12]. At the same time, the kernel coefficients we generated as ground truth values come respectively:

- form the Taylor expansion for the heat kernels

$$g(\hat{\lambda}) \approx e^{-2\lambda} \tag{2.12}$$

- and from one of the synthetic kernels used in the dataset from [10]

(Cri says: ufficializza i parametri in seguito) The number of iterations we apply in this first part of the work is 60, as we found it is a good value for the algorithm to properly converge to a minimum, while the value of ϵ for the spectral boundaries in 2.3 has been set to 0.2. Moreover, we assume the sparsity coefficient T_0 to be equal to 4 and the coherence parameter μ to 10^{-4}.

We use the *sdpt3* solver in the *yalmip* optimization toolbox to solve the quadratic problem and, in order to directly compare the methods here described, we always use the OMP optimization criteria for the sparse coding step.

Finally, the average normalized approximation error we examine for the results quality is defined as **(Cri says: riscrivi la formula a parole tue)**

$$\frac{1}{|Y_{test}|} \sum_{m=1}^{|Y_{test}|} ||Y_m - \mathcal{D}X_m||_2^2 \tag{2.13}$$

with $|Y_{test}|$ being the cardinality of the testing set.

First approach: smoothness in the structure

Second approach: smoothness in the constraints

Results show a general improvement, both regarding the reproduction error and the behavior of the kernel coefficients. The experiments have been repeated several times, in order to be able to claim the stability of the algorithm and show a stable trend. The first important value that testify the improvement of the procedure is the reproduction error, since it decreases significantly when

the smoothness prior is included in the optimization process. In Table 6.1 and Table 6.2 there is a comparison between the error obtained with and without the smoothness assumption, both for the case in which we have one heat kernel and the case in which we used the smooth kernel from the work of Thanou et al.: it can be noticed that its value tends to be stable during several iterations.

Iteration number	Original algorithm	Smoothness assumption
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0

Table 6.1: Reproduction error comparison for the Heat kernel dataset

Iteration number	Original algorithm	Smoothness assumption
1	0	0
2	0	0
3	0	0
4	0	0
5	0	0

Table 6.2: Reproduction error comparison for the low frequency kernel from Thanou et al. dataset

The improvement can then be noticed in the behavior of the kernels' coefficients: in Figure 6.1 there is a comparison between the values assumed by the original heat kernels' coefficients (blue lines), and the ones assumed by the learned coefficients in the case we are applying the smoothness prior or not. It can be seen how the coefficients in the case of no smoothness priors tend to have a less defined trend, while in the smoothness case, they follow the original trend more faithfully. This aspect is shown even more clearly in the case of Thanou et al.'s dataset (Figure 6.2), in which the coefficients have almost the same behavior, apart from their absolute value.

Moreover, Figure 6.3 and Figure 6.4 show the comparison between the original kernel and the

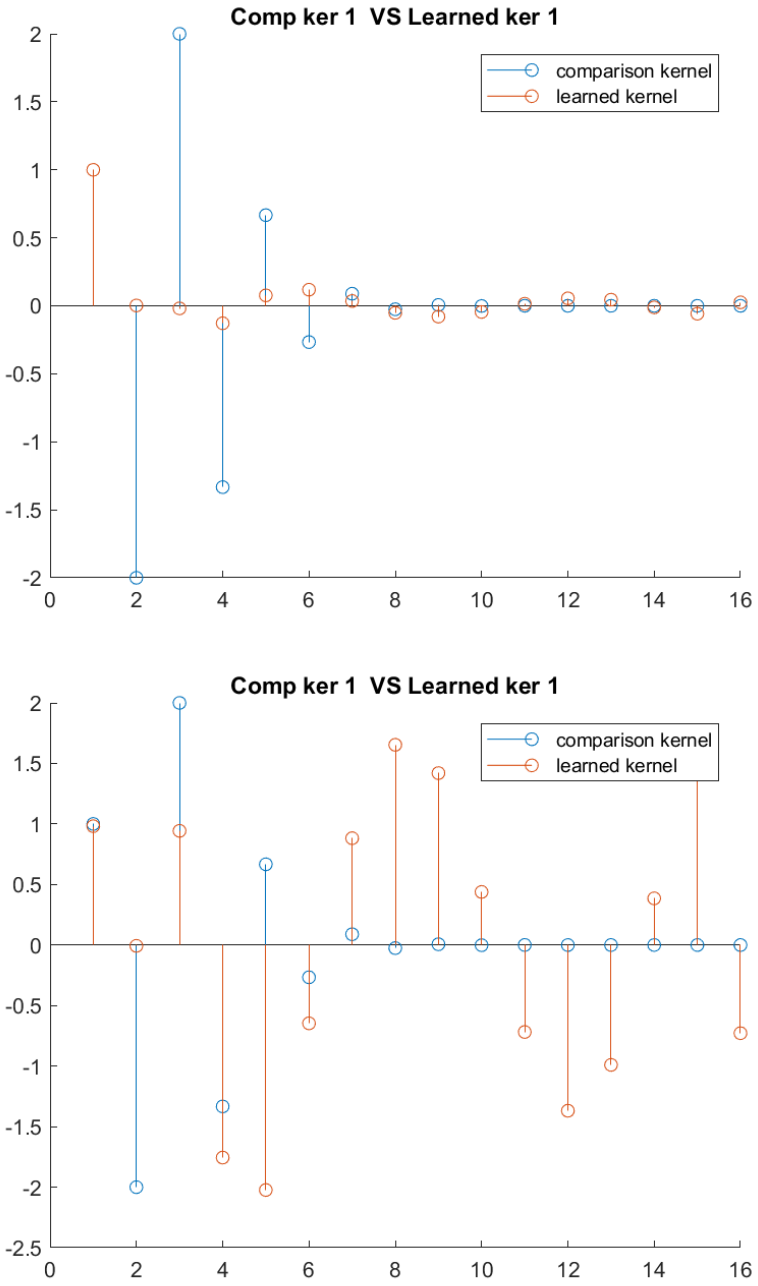


Figure 6.1: Comparison between kernels coefficients without and with smoothness prior. Heat kernel dataset

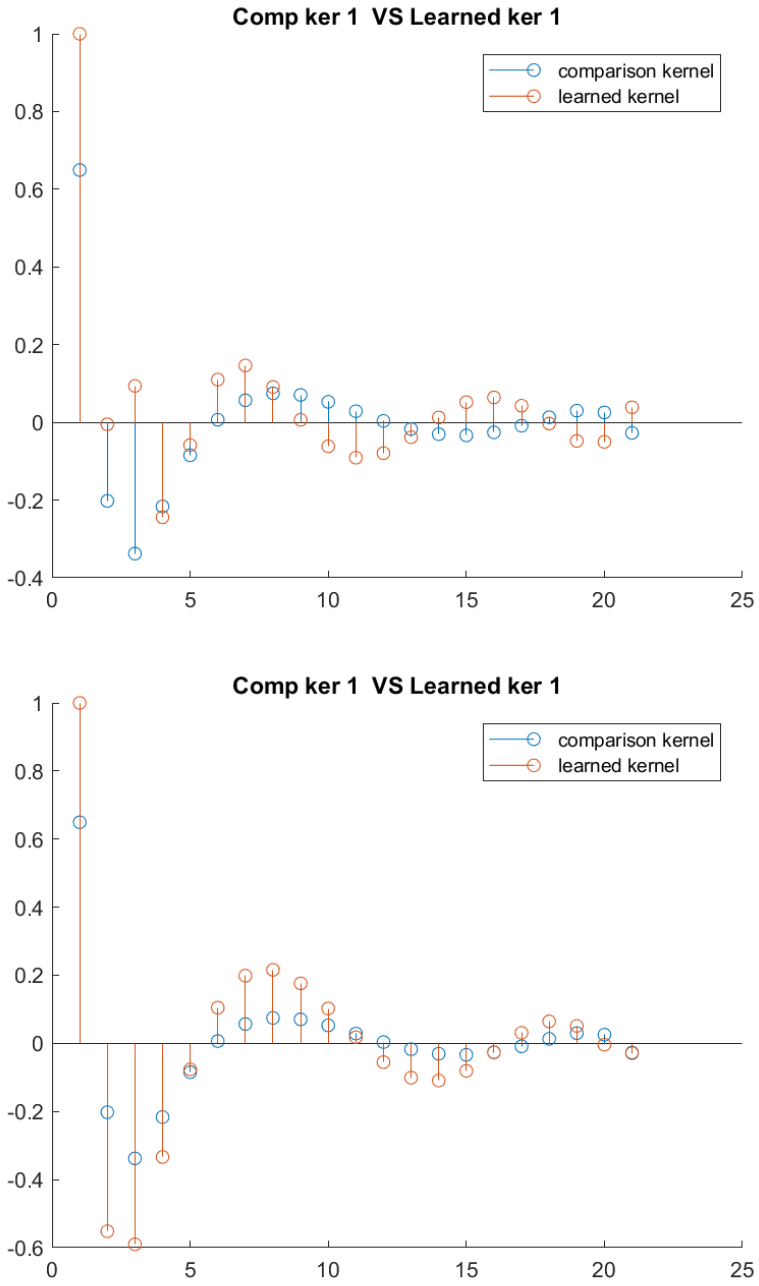


Figure 6.2: Comparison between kernels coefficients without and with smoothness prior. Thanou et al. dataset

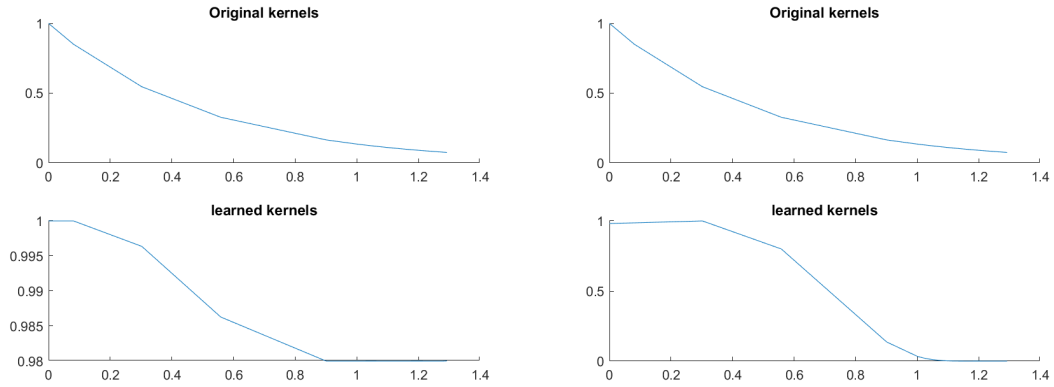


Figure 6.3: Comparison between kernels without and with smoothness prior. Heat kernel dataset

learned kernel without and with smoothness prior in the case of a Heat kernel (6.3) and the dataset from Thanou et al. (6.4). It can be seen how the use of smoothness improves the overall trend also for the kernel shape, which is the ultimate purpose of the algorithm.

Finally, if we look at the behavior of the CPU time necessary for every optimization step, we see that the values in the case we add our prior are in the same order of the ones obtained without it, which gives us further confirmation that out method, despite adding complexity due to the constraints, it does not, though, increase the computational cost.

(Cri says: aggiungi breve descrizione del graph learning)

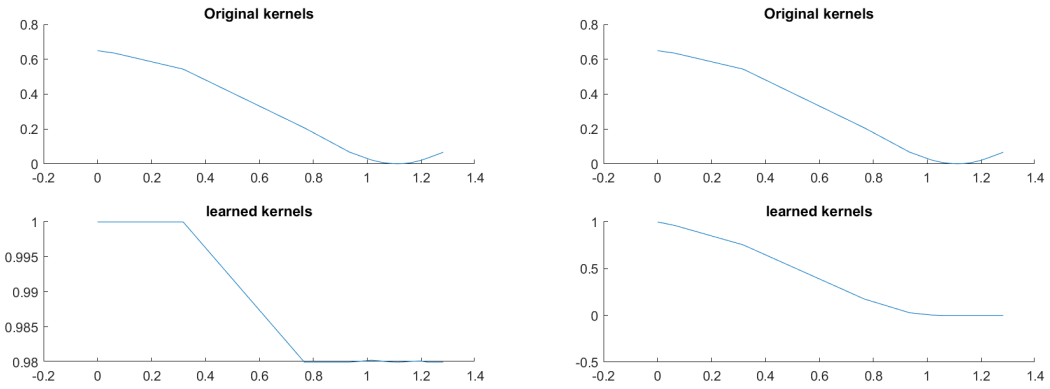


Figure 6.4: Comparison between kernels without and with smoothness prior. Thanou et al. kernel dataset

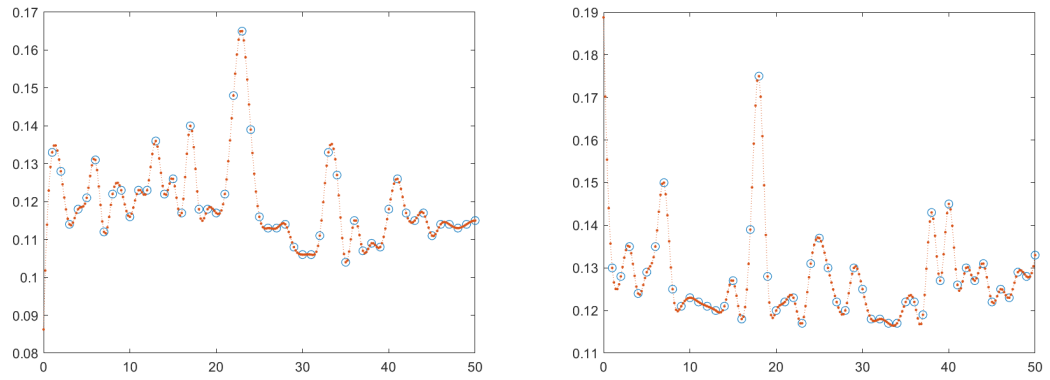


Figure 6.5: Comparison between kernels without and with smoothness prior. Heat kernel dataset

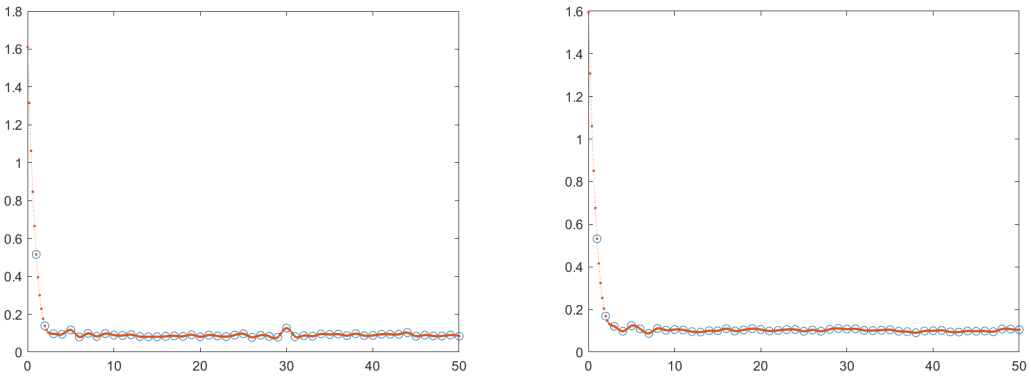


Figure 6.6: Comparison between kernels without and with smoothness prior. Thanou et al. kernel dataset

Chapter 7

Merging the graph and the dictionary learning part

(Cri says: nelle osservazioni aggiungi: - Il criterio di scelta dell'inizailizzazione —> Cerca di capire bene come viene inizializzato il dizionario per essere sicura che quello che stai dicendo nella sezione non è una cazzata - Il fatto che l'inizializzazione non è così vincolante - perchè ci sono disparità nel numero di iterazioni) The next step of our work focuses on the attempt to learn at the same time the Dictionary and the graph structure, while in a further moment we will focus on adding the previous smoothness constraint. As it is formulated theoretically, the double optimization problem could reveal itself to be heavily ill posed, since now the number of parameters involved in the optimization would go up to three instead of the already challenging two. To remedy the problem we decided to create an algorithm which alternates the dictionary learning and the graph learning step, while the sparsity prior is checked at every step by the continuous optimization of the sparsity matrix X.

1. The initialization section

In this situation, one first aspect we have to be careful about is how we initialize the elements for the optimization, namely W and D, since both them are clearly necessary from the beginning. The choice is between initialising the kernels coefficients and initializing the graph structure: since

each entity is strictly correlated with the other, one of the two can be reduced. Our hypothesis is that forcing some kind of structure in the kernels, even though random, could heavily influence the overall behavior of the optimization structure, since a small error in their estimation could easily propagate through the structures that develop onto them; on the other hand, initializing the graph structure in a random way would not bias our learning problem with the same importance. Moreover, the nature of the two learning parts is slightly different: in the dictionary learning part the algorithm attempts to find the only solution that could exist, while in the graph learning part the algorithm aims at finding a good solution for the problem, without necessarily meaning that the graph learned is the same as the ground truth one, since it just has to be one possible solution to the problem. Under this light, it is clear that the graph initialization has smaller consequences than the initialization of the kernels. In order to verify our reasoning we run some tests to see if the intuition was correct and the results aligned with it. **(Cri says: mostrali nei risultati poi)**.

Therefore, in this first part we initialize the weight matrix W, obtain the related normalized Laplacian $\mathcal{L}$ and compute the orthogonal basis connected to it on order to use the eigenvectors we find to have a first estimate on the dictionary.

2. The alternation between optimization steps

With this starting elements we are then able to pass to the alternating optimization steps: every iteration of the global cycle involves the sparsity matrix estimation X through the OMP method (the so-called *sparse coding step*), while what occur in turn every time are the *dictionary learning* and the *graph learning* steps. Finally, after the learning steps there is a re-estimation of the dictionary based on the new values of α and $\mathcal{L}$ which will then be used in the next iteration from the spars coding step. **(Cri says: vedi se puoi spiegarlo un po' meglio)**.

The general algorithm thus articulates in the following way:

Algorithm 1 Parametric dictionary and graph learning

1: **procedure** INITIALIZATION
2: $Y \leftarrow$ Signal samples set
3: $T_0 \leftarrow$ Target sparsity
4: $K \leftarrow$ Polynomial degree
5: $S \leftarrow$ Number of subdictionaries
6: $iterNum \leftarrow$ Number of iterations
7: $W \leftarrow$ Random thresholded adjacency matrix
8: **end procedure**
9: **for** $i = 1, 2, \ldots, iterNum$ **do**
10: **procedure** SPARSITY STEP
11: $X \leftarrow$ Sparsity estimation with OMP
12: **end procedure**
13: **procedure** LEARNING STEP
14: **if** $mod(i, 5) = 0$ **then**
15: $\alpha \leftarrow$ kernels estimation through sdpt3
16: **else**
17: $W \leftarrow$ graph estimation through gradient descent
18: $\mathcal{L} \leftarrow D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$
19: **end if**
20: **end procedure**
21: **procedure** DICTIONARY UPDATE STEP
22: $\mathcal{D} \leftarrow \sum_{k=0}^{K} \alpha_k \mathcal{L}^k$
23: **end procedure**
24: **end for**

We specify that, as shown in the pseudo code at line 14, in our algorithm there is no equal distribution of the iterations dedicated to graph and dictionary learning, but there is a rate $5:1$ in favour of the graph learning. This choice was born from two main facts:

- While the dictionary learning procedure is converging to its minimums discretely fast, the graph learning part requires more steps for it, concluding that giving more space of action to it was a way to balance the optimization steps;
- The sparsity step revealed to be more sensible to alterations of its inputs (the dictionary in our case) when those were due to the kernel coefficients (and so the dictionary learning step) in such a way that limiting these learning steps in their number was advisable;

3. Results

For the tests we run regarding this part we deployed the same datasets used in subsection 2.4: a graph signal with edge weights taken from a gaussian distribution, a Heat kernel and the kernel isolated from Thanou et al. dataset.

Moreover, the number of iterations we chose is $iterNum = 250$, the sparsity factor still equal to $T_0 = 4$ and the coherence factor $\mu = 10^{-4}$.

3.1 first part: the dictionary learning under smoothness constraints

3.2 second part: the graph learning part

3.3 Merging the two approaches

4. further improvements

4.1 graph signal clustering

4.2 varargin

5. conclusions

6. acknoledgements

Bibliography

[1] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst, "Graph Signal Processing: Overview, Challenges and Applications," pp. 1–18, 2017.

[2] A. Sandryhaila, "Signal Processing," no. September, pp. 80–90, 2014.

[3] D. I. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, no. 3, pp. 83–98, 2013.

[4] S. Processing, "Graph learning under sparsity priors Hermina Petric Maretic , Dorina Thanou and Pascal Frossard," pp. 6523–6527, 2017.

[5] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning Laplacian Matrix in Smooth Graph Signal Representations," *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.

[6] I. Tosic and P. Frossard, "Dictionary Learning, What is the right representation for my signal?," *Signal Processing Magazine, IEEE*, vol. 28, no. 2, pp. 27–38, 2011.

[7] B. R. Rubinstein, A. M. Bruckstein, and M. Elad, "Dictionaries for Sparse Representation Modeling.pdf," vol. 98, no. 6, pp. 1045–1057, 2010.

[8] I. F. Gorodnitsky and B. D. Rao, "Sparse signal reconstruction from limited data using FOCUSS: A re-weighted minimum norm algorithm," *IEEE Transactions on Signal Processing*, vol. 45, no. 3, pp. 600–616, 1997.

[9] Y. Pati, R. Rezaiifar, and P. Krishnaprasad, "Orthogonal matching pursuit: recursive function approximation with applications to wavelet decomposition," *Proceedings of 27th Asilomar Conference on Signals, Systems and Computers*, pp. 40–44, 1993.

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

[11] D. I. Shuman, B. Ricaud, and P. Vandergheynst, "Vertex-frequency analysis on graphs," *Applied and Computational Harmonic Analysis*, vol. 40, no. 2, pp. 260–291, 2016.

[12] V. Kalofolias, "How to learn a graph from smooth signals," vol. 2, 2016.

[13] ""Approximating signals supported on graphs"," *Signals, Department of Electrical and Computer Engineering , McGill University 3480 University St , Montreal QC , Canada H3A 2A7*, vol. 3, pp. 3921–3924, 2012.

[14] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Transactions on Information Theory*, vol. 50, no. 10, pp. 2231–2242, 2004.