Master Project title

Final master project

Cristina Gava

July 10, 2018

Contents

1. Introduction: 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]

1.1 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 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 shows an example of graph signal: the colours of the vertices represents the values the signal has on them.

1.2 the inverse covariance matrix

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

1.2.1 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

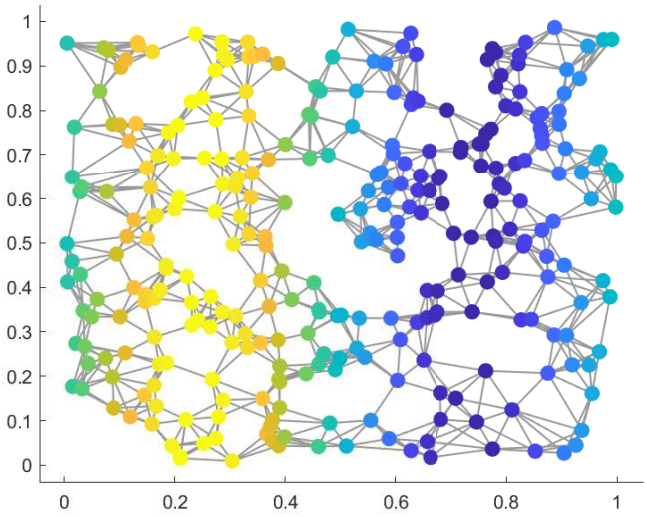

Figure 1: Random sensor graph

vertex i. This entity is a difference operator, since it satisfies the condition:

$$((Lf)(i) = \sum_{j \in \mathcal{N}_i} W_{i,j}[f(i) - f(j)] \tag{1.1}$$

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 $\{u_l\}_{l=0,1,\ldots,N-1}$, to which real and non-negative eigenvalues are associated ($\{\lambda_l\}_{l=0,1,\ldots,N-1}$). Together with the eigenvectors, they satisfy the condition:

$$Lu_l = \lambda_l u_l, \text{ for } l = 0, 1, \ldots, N-1 \tag{1.2}$$

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 $\sigma(L) = \{\lambda_0, \lambda_1, \ldots, \lambda_{N-1}\}$ 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:

$$\hat{f}(\xi) = \langle f, e^{2\pi i \xi t} \rangle = \int_{\mathbb{R}} f(t) e^{-2\pi i \xi t} dt \tag{1.3}$$

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:

$$-\Delta(e^{2\pi i \xi t}) = -\frac{\partial^2}{\partial t^2} e^{2\pi i \xi t} = (2\pi\xi)^2 e^{2\pi i \xi t} \tag{1.4}$$

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]:

$$\hat{f}(\lambda_l) = \langle f, u_l \rangle = \sum_{i=1}^{N} f(i) u_l * (i) \tag{1.5}$$

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

$$f(i) = \sum_{l=0}^{N-1} \hat{f}(\lambda_l) u_l(i) \tag{1.6}$$

2. 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. [4] 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 over-completeness 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. [4] [5]

With these premises, we consider a dictionary $\mathbf{D} = [\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_L] \in \mathbb{R}^{N \times L}$, 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:

$$\mathbf{x} = \mathbf{D}^T \gamma_s \tag{2.7}$$

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

$$\gamma_a = \mathbf{D}^T \mathbf{x} \tag{2.8}$$

The representation in 2.7 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:

$$\gamma_s = \underset{\gamma}{\operatorname{argmin}}\ \mathcal{C}(\gamma) \quad \text{Subject To } \mathbf{x} = D\gamma \tag{2.9}$$

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 2.9 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.

2.1 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. [6] [7] 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. [5] **(Cri says: forse aggiungi parte sugli sparse dictionaries sempre presa dallo stesso paper. Poi: vedi se aggiungere un excursus storico sui dizionari ad apprendimento)**.

2.1.1 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:

$$\mathbf{y} = \phi\mathbf{a} = \sum_{k=1}^{N} a_k \phi_k \tag{2.10}$$

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:

$$\min_{\mathbf{a}} \ (||\mathbf{y} - \phi\mathbf{a}||_2^2 + \lambda||\mathbf{a}||_1) \tag{2.11}$$

in which the relaxation allowed to replace the nonconvex l^0 norm in the original problem with a more meaningful l^1 norm. [4]

2.2 The correspondence between the dictionary and the graph signal

In 1.2.1 we introduced the Laplacian operator as $L = D - W$, here we focus on a modified version of this operator, that is the *normalized Laplacian* $\mathcal{L} = D^{-\frac{1}{2}} L D^{-\frac{1}{2}}$, 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:

$$\hat{y}(\lambda_l) = \langle y, \chi_l \rangle = \sum_{n=1}^{N} y(n) \chi_l^*(n) \tag{2.12}$$

and its relative inverse transform to be:

$$y(n) = \sum_{l=0}^{N-1} \hat{y}(\lambda_l) \chi_l(n), \quad \forall n \in \mathcal{V} \tag{2.13}$$

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] [8]:

$$T_n g = \sqrt{N}(g * \delta_n) = \sqrt{N} \sum_{l=0}^{N-1} \hat{g}(\lambda_l)\chi_l^*(n)\chi_l \tag{2.14}$$

this leading to think about 2.14 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 2.14 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 2.14 is a smooth polynomial function of degree K:

$$\hat{g}(\lambda_l) = \sum_{k=0}^{K} \alpha_k \lambda_l^k, \quad l = 0, \ldots, N-1 \tag{2.15}$$

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

$$T_n g = \sqrt{N}(g * \delta_n) = \sqrt{N} \sum_{l=0}^{N-1} \sum_{k=0}^{K} \alpha_k \lambda_l^k \chi_l^*(n)\chi_l \tag{2.16}$$

$$= \sqrt{N} \sum_{k=0}^{K} \alpha_k \sum_{l=0}^{N-1} \lambda_l^k \chi_l^*(n)\chi_l = \sqrt{N} \sum_{k=0}^{K} \alpha_k (\mathcal{L}^k)_n \tag{2.17}$$

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

$$Tg = \sqrt{N}\hat{g}(\mathcal{L}) = \sqrt{N}\chi\hat{g}(\Lambda)\chi^T = \sqrt{N} \sum_{k=0}^{K} \alpha_k \mathcal{L}^k \tag{2.18}$$

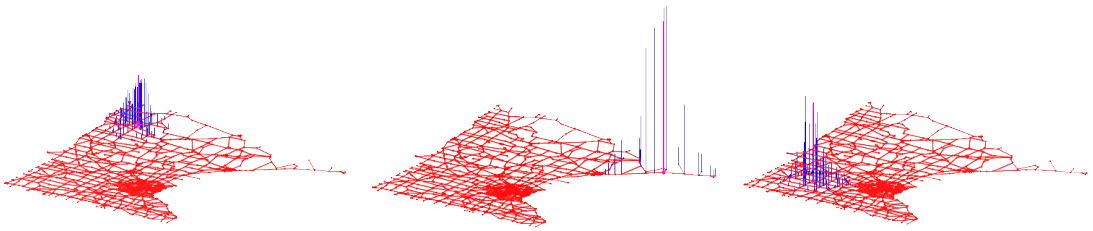

Figure 2: Example of graph signal translation

Where the quantity Λ corresponds to the diagonal matrix of the eigenvalues in such a way that the following relation holds: $\mathcal{L} = \chi\Lambda\chi^T$ [9]. 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 there is an example of an atom translation in three different ways, $T_{100}g$, $T_{200}g$ and $T_{2000}g$.

2.3 The smoothness assumption

citare [10]

3. Brief state of the art

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

5. Problem presentation

6. main improvements

6.1 first part: the dictionary learning under smoothness constraints

6.2 second part: the graph learning part

6.3 Merging the two approaches

7. further improvements

7.1 graph signal clustering

7.2 varargin

8. conclusions

9. acknoledgements

References

[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] 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.

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

[6] 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.

[7] 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.

[8] 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.

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

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