

Learning Laplacian Matrix from Graph Signals with Sparse Spectral Representation

Pierre Humbert, Batiste Le Bars, Laurent Oudre, Argyris Kalogeratos, Nicolas Vayatis

▶ **To cite this version:**

Pierre Humbert, Batiste Le Bars, Laurent Oudre, Argyris Kalogeratos, Nicolas Vayatis. Learning Laplacian Matrix from Graph Signals with Sparse Spectral Representation. Journal of Machine Learning Research, 2021. hal-03671262

HAL Id: hal-03671262
https://hal.science/hal-03671262

Submitted on 18 May 2022

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Journal of Machine Learning Research 22 (2021) 1-47 Submitted 11/19; Revised 4/21; Published 10/21

Learning Laplacian Matrix from Graph Signals with Sparse Spectral Representation

Pierre Humbert[1] PIERRE.HUMBERT@ENS-PARIS-SACLAY.FR
Batiste Le Bars[1] BATISTE.LEBARS@ENS-PARIS-SACLAY.FR
Laurent Oudre LAURENT.OUDRE@ENS-PARIS-SACLAY.FR
Argyris Kalogeratos ARGYRIS.KALOGERATOS@ENS-PARIS-SACLAY.FR
Nicolas Vayatis NICOLAS.VAYATIS@ENS-PARIS-SACLAY.FR

Centre Borelli, ENS Paris-Saclay, Université Paris-Saclay, CNRS, F-94235 Cachan, France.

Editor: Tina Eliassi-Rad

Abstract

In this paper, we consider the problem of learning a graph structure from multivariate signals, known as *graph signals*. Such signals are multivariate observations carrying measurements corresponding to the nodes of an unknown graph, which we desire to infer. They are assumed to enjoy a sparse representation in the graph spectral domain, a feature which is known to carry information related to the cluster structure of a graph. The signals are also assumed to behave smoothly with respect to the underlying graph structure. For the graph learning problem, we propose a new optimization program to learn the Laplacian of this graph and provide two algorithms to solve it, called IGL-3SR and FGL-3SR. Based on a 3-step alternating procedure, both algorithms rely on standard minimization methods –such as manifold gradient descent or linear programming– and have lower complexity compared to state-of-the-art algorithms. While IGL-3SR ensures convergence, FGL-3SR acts as a relaxation and is significantly faster since its alternating process relies on multiple closed-form solutions. Both algorithms are evaluated on synthetic and real data. They are shown to perform as good or better than their competitors in terms of both numerical performance and scalability. Finally, we present a probabilistic interpretation of the proposed optimization program as a Factor Analysis Model.

Keywords: Graph learning, graph Laplacian, non-convex optimization, graph signal processing, sparse coding, clustering

1. Introduction

Hidden structures in multivariate or multimodal signals can be captured through the notion of *graph*. The availability of such a graph is a core assumption in many computational tasks such as spectral clustering, semi-supervised learning, graph signal processing, etc. However, in most situations no natural graph can be derived or defined, therefore the underlying graph must be inferred from available data. This task, often referred to as *graph learning*, has

1. Authors with equal contribution.

©2021 Pierre Humbert, Batiste Le Bars, Laurent Oudre, Argyris Kalogeratos, and Nicolas Vayatis.

License: CC-BY 4.0, see https://creativecommons.org/licenses/by/4.0/. Attribution requirements are provided at http://jmlr.org/papers/v22/19-944.html.

received significant attention in fields such as machine learning, signal processing, biology, meteorology, and others (Friedman et al., 2008; Hecker et al., 2009; William et al., 2017).

Learning a graph is an ill-posed problem as several graphs can explain the same set of observations. Previous works have been devoted to introducing models or constraints that would narrow down the range of possible solutions. For instance, physical constraints can be imposed to suggest epidemic models or other information propagation and interaction models (Rodriguez et al., 2011; Du et al., 2012; Gomez-Rodriguez et al., 2016). From a statistical perspective, the graph learning task is seen as the estimation of a certain probability distribution parametrized by the graph itself. Generally, the assumed class of distributions is either a *Bayesian Network* in the case of directed graphs, or a *Markov Random Field* for undirected graphs (Koller et al., 2009; Yang et al., 2015; Wang and Kolar, 2016; Tarzanagh and Michailidis, 2018; Le Bars et al., 2020). Hence, the graph structure encompasses the conditional dependencies between variables. Two variables will be connected in the graph if they are dependent conditionally on all the other variables. In the particular case of Gaussian Random Fields, the graph estimation consists in the estimation of the inverse covariance matrix, known as the *precision matrix* (Banerjee et al., 2008; Friedman et al., 2008). In Friedman et al. (2008), the proposed estimation method corresponds to the well-known Graph-Lasso algorithm, which relies on the assumption that the precision matrix is subject to a sparsity constraint.

More recently, *Graph Signal Processing* (GSP) (Shuman et al., 2013; Djuric and Richard, 2018), has generalized the standard concepts and tools of signal processing to multivariate signals recorded over graph structures. Notions such as smoothness, sampling, filtering, etc., have been adapted to this framework, opening a new field that paves the way to further developments in graph learning (Pasdeloup et al., 2017; Thanou et al., 2017). In this framework, the *smoothness* of the observations with respect to the true underlying graph is a common assumption (Dong et al., 2018; Daitch et al., 2009; Kalofolias, 2016; Egilmez et al., 2016; Chepuri et al., 2017), which asks for graphs on which the signals have small local variations among adjacent nodes. Another naturally arising property of real-world problems is the sparsity of the observations in the graph spectral basis (Valsesia et al., 2018; Sardellitti et al., 2019). In data clustering, for instance, the vector of labels seen as a signal over the vertices of a graph, exhibits a sparse spectral representation. It is smooth within each cluster and varies across different clusters (Figure 1). Hence, building this kind of graph is relevant for graph-based clustering approaches such as spectral clustering. Furthermore, the sparsity assumption is also relevant for the sampling task. Indeed, by making use of this property, it is possible under mild conditions to reconstruct the dimensions of the multivariate observations corresponding to nodes that have not been sampled (Chen et al., 2015). These properties, all borrowed from the GSP field, can be seen as constraints that can be used as regularizations to the graph learning task, and offer a new perspective on the topic.

Aim and main contribution. In the present paper, we introduce an optimization problem to learn a graph from signals that are assumed to be smooth and admitting a sparse representation in the spectral domain of the graph. The main axes of our contribution can be summarized as follows:

- The graph learning task problem is cast as the optimization of a smooth nonconvex objective function over a nonconvex set (Section 2). This problem is efficiently solved by introducing a framework that combines in an innovative way barrier methods, alternating minimization, and manifold optimization (Section 3). A relaxed algorithm is also proposed, which is scalable with respect to the graph dimensions (Section 4).

- A factor analysis model for smooth graph signals with sparse spectral representation is introduced (Section 5). This model provides a probabilistic interpretation of our optimization program and links its objective function to a maximum a posteriori estimation.

- The proposed algorithms are tested and compared to state-of-the-art approaches using several synthetic and real datasets (Section 7). The experimental results show that our approach achieves similar or better performance than existing methods, while reducing significantly the need of computing resources.

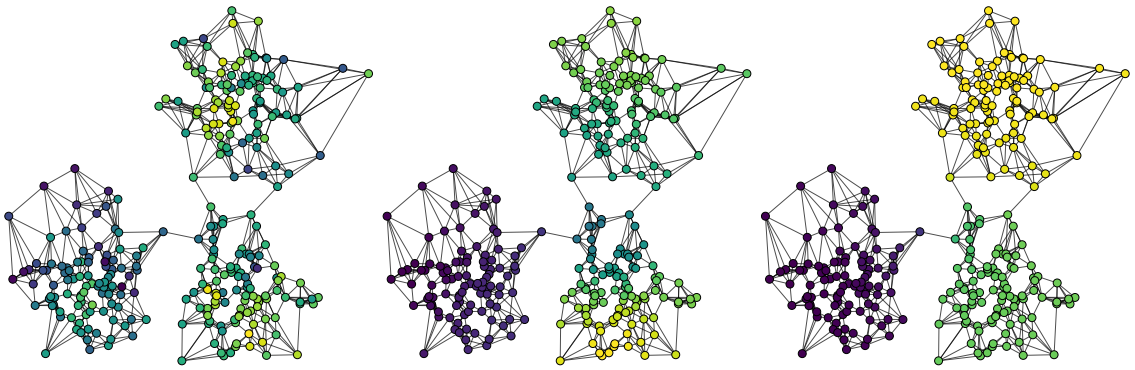
Background and notations. Throughout the paper, we consider an undirected and weighted graph G with no self-loops. It is defined as a pair $G = (\mathcal{V}, \mathcal{E})$ with vertices (or nodes) $\mathcal{V} = \{1, ..., N\}$ and set of edges $\mathcal{E} = \{(i, j, w_{ij}), i, j \in \mathcal{V}\}$ with weights $w_{ij} \in \mathbb{R}_+$ arranged in a weight matrix $W \in \mathbb{R}_+^{N \times N}$. More particularly, we focus on its *combinatorial graph Laplacian* matrix that describes entirely the graph and is given by $L = D - W$, where D is the diagonal degree matrix and W the weight matrix. As G is undirected, L is a symmetric positive semi-definite matrix. Its eigenvalue decomposition can be written as $L = X \Lambda X^\mathsf{T}$, with $\Lambda = \mathrm{diag}(\lambda_1, ..., \lambda_N)$ a diagonal matrix with the eigenvalues and $X = (x_1, ..., x_N)$ a matrix with the eigenvectors as columns. We also consider graph signals (or graph functions) on this graph. A graph signal is defined as a function $y : \mathcal{V} \to \mathbb{R}^N$ that assigns a scalar value to each vertex. This function can be represented as a vector $y \in \mathbb{R}^N$, with y_i the function value at the i-th vertex. Also, with $\mathbf{1}_N$ we denote the constant unitary vector of size N, and with $\mathbf{0}_N$ the vector containing only zeros. Additional notations are given when needed in the text, while all our notations are collected in Table 3 (Section 9).

Next, we give important definitions for two graph signal properties: first the *smoothness*, and then the *spectral sparsity* that allows us to create a spectral representation of a graph signal y adapted to a graph, using the Graph Fourier Transform (GFT).

Definition 1 (Smoothness) – *Let $G = (\mathcal{V}, \mathcal{E})$ be a graph, L be its Laplacian matrix, and $y \in \mathbb{R}^N$ be a graph signal seen as vector. Given a smoothness level $s \geq 0$, then a graph signal y is said to be s-smooth with respect to the graph G if*

$$y^\mathsf{T} L y = \frac{1}{2} \sum_{i,j} w_{ij} \left(y_i - y_j\right)^2 \leq s \; . \tag{1}$$

Intuitively, a graph signal y is s-smooth with respect to G if adjacent nodes of the graph carry sufficiently similar signal values.



(a) **(b)** **(c)**

Figure 1: Three smooth graph signals ($N = 300$) with decreasing bandlimitedness, each of them admitting a spectral representation that is: (a) a 150-sparse, (b) 6-sparse, and (c) 3-sparse.

Definition 2 (Graph Fourier Transform) – *Let $G = (\mathcal{V}, \mathcal{E})$ be an undirected graph with no self-loops, and $L = X\Lambda X^\mathsf{T}$ be the eigenvalue decomposition of its Laplacian matrix. Then, the GFT of a graph signal $y \in \mathbb{R}^N$ is given by $h = X^\mathsf{T} y$, where the components of h are interpreted as Fourier coefficients, the eigenvalues Λ as distinct frequencies, and the eigenvectors X as decomposition basis.*

Definition 3 (Spectral sparsity) – *Consider the notations of Definition 2. Given $k \in \mathbb{N}^+$, we say that a graph signal y admits a k-sparse spectral representation with respect to a graph G, if for $h = X^\mathsf{T} y$, then*

$$\|h\|_0 \leq k \;, \tag{2}$$

where $\|h\|_0$ stands for the number of non-zero elements of h.

Regarding this definition, y admits k-sparse spectral representation if the number of non-zero elements in its Fourier coefficients vector is less or equal to k. In the following, we also use the term *k-bandlimited* when referring to a k-sparse signal.

2. Problem Statement

2.1 Setup and working assumptions

The general task of *graph learning* aims at inferring a graph G that best explains the n observed graph signals $\{y^{(i)}\}_{i=1}^n$ of size N, composing a matrix $Y = [y^{(1)}, \cdots, y^{(n)}] \in \mathbb{R}^{N \times n}$. The proposed graph learning framework takes as input the matrix Y and outputs the Laplacian matrix L associated to the inferred graph G (note that both notions are equivalent). Our learning process is based on the following assumptions:

Assumption 1 (Assumption on the graph G) – *G is undirected, with no self-loops.*

With Assumption 1, L is a symmetric positive semi-definite matrix with eigenvalue decomposition $L = X\Lambda X^\mathsf{T}$, where $\lambda_1 = 0$ and $x_1 = \frac{1}{\sqrt{N}}\mathbf{1}_N$ (Chung and Graham, 1997).

Assumption 2 (Assumption on the signals Y) – *Graph signals Y defined over the true underlying graph G are assumed s-smooth and admit a k-sparse spectral representation, with unknown values for s and k.*

On the smoothness assumption. According to Equation (1), low s values tend to favor smooth signals for which adjacent nodes carry similar signal values. This natural property has been widely considered in graph learning (Daitch et al., 2009; Dong et al., 2016).
On the spectral sparsity assumption. This property is known as *bandlimitedness* in the GSP field and assumes that the null components of h are those associated to the largest eigenvalues (frequencies). Essentially, this additional hypothesis expresses a fundamental principle of signal processing that suggests filtering-out the high-frequency band of a signal, as it carries mainly noise and little or no information. This assumption is very common for graph signals, especially in GSP where it is the main hypothesis of several graph sampling methods (Anis et al., 2014; Narang et al., 2013; Chen et al., 2015; Marques et al., 2016). This property is also related to the cluster structure of a graph. Indeed, in a graph of k clusters, a signal that is smooth within each cluster and vary arbitrarily across clusters, will admit a k-sparse spectral representation. In this context, the non-null weights of h will be necessarily associated to the k first eigenvectors of the corresponding Laplacian matrix, as these eigenvectors are also smooth within the clusters (Von Luxburg, 2007). To enforce such behavior in the graph learning process, i.e. make sure that only the first coefficients of h are non-zero, the bandlimitness property must be combined with the smoothness property. Figure 1 shows three smooth graph signals to illustrate the intuition behind these assumptions.

2.2 Graph Learning for Smooth and Sparse Spectral Representation

A general graph learning scheme consists in learning the adjacency or the Laplacian matrix. However, since the constraint of Assumption 2 (sparsity of the graph signals over the eigenbasis of the Laplacian matrix) is easier to be expressed in the spectral domain, in this article we focus on learning the eigendecomposition of the Laplacian matrix $L = X\Lambda X^\mathsf{T}$. The optimization problem incorporates a linear least square regression term depending of Y, X, and H, which controls the distance of the new representation XH to the observations Y. In addition, due to Assumption 2, we add two penalization terms: One to control the smoothness of the new representation, depending on Λ and H; the other one to control it's sparsity on the spectral domain, which only depends on H. Finally, as we want to learn a Laplacian matrix satisfying Assumption 1, equality and inequality constraints relative to X and Λ are necessary. To that end, we introduce the following optimization problem:

$$\min_{H,X,\Lambda} \|Y - XH\|_F^2 + \alpha\|\Lambda^{1/2}H\|_F^2 + \beta\|H\|_S \;, \tag{3}$$

$$\text{s.t.} \quad \begin{cases} X^\mathsf{T}X = I_N,\ x_1 = \frac{1}{\sqrt{N}}\mathbf{1}_N \;, & \text{(a)} \\ (X\Lambda X^\mathsf{T})_{k,\ell} \leq 0 \quad k \neq \ell \;, & \text{(b)} \\ \Lambda = \text{diag}(0, \lambda_2, ..., \lambda_N) \succeq 0 \;, & \text{(c)} \\ \text{tr}(\Lambda) = N \in \mathbb{R}_*^+ \;, & \text{(d)} \end{cases}$$

where $\|\cdot\|_S$ is defined bellow, I_N is the identity matrix of size N, $\text{tr}(\cdot)$ denotes the trace of an input matrix, and $\Lambda \succeq 0$ indicates that the matrix Λ is semi-definite positive.

This problem aims at conjointly learning the Laplacian L (i.e. (X, Λ)) and a smooth bandlimited approximation XH of the observed signals Y. Here, H has the same size as Y and corresponds to the spectral representation of the graph signals through the GFT.

Interpretation of the terms. In the objective function (3), the first term corresponds to the quadratic approximation error of Y by XH, where $\|\cdot\|_F$ is the Frobenius norm. The second term is a *smoothness regularization* equally imposed to each column of XH. Indeed, from Equation (1), we have $\sum_i y^{(i)\mathsf{T}} L y^{(i)} = \mathrm{tr}(Y^\mathsf{T} L Y) = \|L^{1/2} Y\|_F^2$. Rewriting this for the set of graph signals in XH, we obtain:

$$\|L^{1/2} XH\|_F^2 = \|X\Lambda^{1/2} X^\mathsf{T} XH\|_F^2 = \|\Lambda^{1/2} H\|_F^2 = \sum_{i=1}^{N} \lambda_i \|H_{i,:}\|_2^2 \,,$$

where $H_{i,:}$ is the i-th row of the matrix H. This kind of regularization is very common in graph learning (Kalofolias, 2016; Chepuri et al., 2017). From its definition, we can see that it tends to be low when high values of $\{\lambda_i\}_{i=1}^N$ are associated to rows of H with low ℓ_2-norm. This corroborates the idea that the $\{\lambda_i\}_{i=1}^N$ can be interpreted as frequencies and the elements of H as Fourier coefficients.

Finally, $\beta\|H\|_S$, is a *sparsity regularization*. In this work, we propose to either use the $\ell_{2,1}$ (the sum of the ℓ_2-norm of each row of H) or $\ell_{2,0}$ (the number of rows with ℓ_2-norm different than 0) that induces a row-sparse solution $\widehat{H}$.

Remark on the choice of $\|\cdot\|_S$. In the context of GSP, it is natural to assume that the graph signals are bandlimited, all at the same dimensions. This property is enforced by $\|\cdot\|_S$ and has two main advantages: it is a key assumption for sampling over a graph and this particular structure is better for inferring graphs with clusters (Sardellitti et al., 2019). Therefore, in this article, the use of the classical ℓ_0-norm and the ℓ_1-norm has not been investigated since they would impose sparsity at every dimension of the matrix H 'independently', which would consequently break the bandlimitedness assumption.

The hyperparameters, $\alpha, \beta > 0$ are controlling respectively the smoothness of the approximated signals and the sparsity of H. A discussion on the influence of these hyperparameters and an efficient way to fix them is provided in Section 7.3.1. Finally, the first three constraints (3a), (3b), (3c) enforce $X\Lambda X^\mathsf{T}$ to be a Laplacian matrix of an undirected graph (Assumption 1). More specifically, by definition, $L = D - W$ with $W \in \mathbb{R}_+^{N\times N}$, thus we necessary have $\forall k \neq \ell, L_{k,\ell} = (X\Lambda X^\mathsf{T})_{k,\ell} \leq 0$ (constraint (3b)). Furthermore, as $X\Lambda X^\mathsf{T}$ is the eigendecomposition of the Laplacian matrix of an undirected graph, $X^\mathsf{T} X = I_N, x_1 = \frac{1}{\sqrt{N}}\mathbf{1}_N$ and$\lambda_1 = 0 \leq \lambda_2 \leq ... \leq \lambda_N$ (constraints (3a) and (3c)). The last constraint (3d) was proposed in Dong et al. (2016) as to impose structure in the learned graph while avoiding that the trivial solution $\widehat{\Lambda} = \mathbf{0}$. A discussion about values other than N is made in Kalofolias (2016).

The objective function (3) is not jointly convex but when $\|\cdot\|_S$ is taken to be the $\ell_{2,1}$ norm, it is convex with respect to each of the block-variables H, X, or Λ, taken independently. A natural approach to solve this problem is therefore to alternate between the three variables, minimizing over one while keeping the others fixed. However, due to the equality constraint

(3a) and inequalities (3b), the feasible set is not convex with respect to X. Hence, this approach raises several difficulties that will be discussed and handled in the following section.

2.3 Reformulation of the problem

As stated in Section 2.2, Problem (3) is not jointly convex and cannot be solved easily with constraints (3a) and (3b). In this section, we propose to rewrite constraints (3a) and (3b), in order to define a new equivalent optimization problem that can be solved with well-known techniques.

2.3.1 Reformulation of the constraint (3*a*)

In this section, we show that the constraints (3a) can be reformulated as a constraint over the space of orthogonal matrices in $\mathbb{R}^{(N-1)\times(N-1)}$. Although such transformation does not change the convexity of the feasible set, we will see in Section 3.3 that there exist efficient algorithms that perform optimization over such manifold.

Definition 4 (Orthogonal group) – *The space of orthogonal matrices in $\mathbb{R}^{N\times N}$, called orthogonal group, is the space:*

$$Orth(N) = \{X \in \mathbb{R}^{N\times N} \mid X^\mathsf{T}X = I_N\} \,.$$

Lemma 5 – *Given $X, X_0 \in \mathbb{R}^{N\times N}$ two orthogonal matrices, both having their first column equal to $\frac{1}{\sqrt{N}}\mathbf{1}_N$ (constraint (3a)), we have the following equality*

$$X = X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & [X_0^\mathsf{T}X]_{2:,2:} \end{bmatrix},$$

with $[X_0^\mathsf{T}X]_{2:,2:}$ denoting the submatrix of $X_0^\mathsf{T}X$ containing everything but the first row and column of itself. Furthermore, $[X_0^\mathsf{T}X]_{2:,2:}$ is in $Orth(N-1)$.

The above lemma allows us to build an equivalent formulation of Problem (3) given by the following proposition.

Proposition 6 – *Given $X_0 \in \mathbb{R}^{N\times N}$ an orthogonal matrix with first column being equal to $\frac{1}{\sqrt{N}}\mathbf{1}_N$, an equivalent formulation of optimization Problem (3) is given by*

$$\min_{H,U,\Lambda} \left\| Y - X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U \end{bmatrix} H \right\|_F^2 + \alpha\|\Lambda^{1/2}H\|_F^2 + \beta\|H\|_S \triangleq f(H,U,\Lambda)\,, \tag{4}$$

$$\text{s.t.} \quad \begin{cases} U^\mathsf{T}U = I_{N-1}\,, & \text{(a')} \\ \left(X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U \end{bmatrix} \Lambda \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U^\mathsf{T} \end{bmatrix} X_0^\mathsf{T} \right)_{k,\ell} \leq 0 \quad k \neq \ell\,, & \text{(b')} \\ \Lambda = \mathrm{diag}(0, \lambda_2, ..., \lambda_N) \succeq 0\,, & \text{(c)} \\ \mathrm{tr}(\Lambda) = N \in \mathbb{R}_*^+\,. & \text{(d)} \end{cases}$$

The latter proposition says that since the first column of X is fixed and known, it is sufficient to look for an optimal rotation of a valid matrix X_0 that preserves the first column. Such

a rotation matrix is given above and is parametrized by a U in $Orth(N-1)$. Note that in practice, to find a matrix X_0 satisfying (3a), we build the Laplacian of any graph with a single connected component and take its eigenvectors.

2.3.2 Log-barrier method for constraint (4b')

In order to deal with constraint (4b'), we propose to use a log-barrier method. This barrier function allows us to consider an approximation of Problem (4) where the inequality constraint (4b') is made implicit in the objective function. Denoting by $f(\cdot)$ the objective function of (4), we want to solve

$$\min_{H,U,\Lambda} \quad f(H,U,\Lambda) + \frac{1}{t}\phi(U,\Lambda) \qquad \text{s.t.} \quad (4a'),(4c),(4d)\ , \tag{5}$$

where t is a fixed positive constant and $\phi(\cdot)$ is the log-barrier function associated to the constraint (4b').

Definition 7 (Log-barrier function) – *Let the following matrix in* $\mathbb{R}^{N\times N}$*:*

$$h(U,\Lambda) = X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U \end{bmatrix} \Lambda \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U \end{bmatrix}^\mathsf{T} X_0^\mathsf{T}\ ,$$

involved in the constraint $(4b')$*. The associated log-barrier function* $\phi : \mathbb{R}^{(N-1)\times(N-1)} \times \mathbb{R}^{N\times N} \longrightarrow \mathbb{R}$ *is defined by:*

$$\phi(U,\Lambda) = -\sum_{k=1}^{N-1}\sum_{\ell>k}^{N} \log\Big(-h(U,\Lambda)_{k,\ell}\Big)\ , \tag{6}$$

with $\mathbf{dom}(\phi) = \big\{(U,\Lambda) \in \mathbb{R}^{(N-1)\times(N-1)} \times \mathbb{R}^{N\times N} \mid \forall 1 \le k < \ell \le N, h(U,\Lambda)_{k,\ell} < 0\big\}$, i.e. its domain is the set of points that strictly satisfy the inequality constraints (4b').

This barrier function allows us to perform block-coordinate descent on three subproblems that are easier to solve, as we discuss in the next section.

3. Resolution of the problem: the IGL-3SR algorithm

In this section, we describe our method, the *Iterative Graph Learning for Smooth and Sparse Spectral Representation* (IGL-3SR), and its different steps to solve Problem (5). Given a fixed $t > 0$, we propose to use a block-coordinate descent on H, U, and Λ, which permits to split the problem in three partial minimizations that we discuss in this section. One of the main advantages of IGL-3SR is that each subproblem can be solved efficiently and as the objective function is lower-bounded over the set of constraints, this procedure ensures convergence. The summary of the method is presented in Algorithm 1.

3.1 Optimization with respect to H

For fixed U and Λ, the minimization Problem (5) with respect to H is:

$$\min_H \|Y - XH\|_F^2 + \alpha\|\Lambda^{1/2}H\|_F^2 + \beta\|H\|_S\ , \quad \text{where } X = X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U \end{bmatrix}. \tag{7}$$

When $\|\cdot\|_S$ is set to $\|\cdot\|_{2,0}$ (resp. $\|\cdot\|_{2,1}$), this problem is a particular case of what is known as Sparsify Transform Learning (Ravishankar and Bresler, 2012) (resp. is a particular case of the Group Lasso (Yuan and Lin, 2006) known as Multi-Task Feature Learning (Argyriou et al., 2007)). Moreover, as X is orthogonal, we are able to find closed-form solutions (Proposition 8).

Proposition 8 (Closed-form solution for the $\ell_{2,0}$ and $\ell_{2,1}$-norms) – *The solutions of Problem* (7) *when* $\|\cdot\|_S$ *is set to* $\|\cdot\|_{2,0}$ *or* $\|\cdot\|_{2,1}$*, are given in the following.*

- *Using the* $\ell_{2,0}$*-norm, the optimal solution of* (7) *is given by the matrix* $\widehat{H} \in \mathbb{R}^{N\times n}$ *where for* $1 \leq i \leq N$,

$$\widehat{H}_{i,:} = \begin{cases} 0 & \text{if } \frac{1}{1+\alpha\lambda_i}\|(X^\mathsf{T}Y)_{i,:}\|_2^2 \leq \beta \ , \\ \frac{1}{(1+\alpha\lambda_i)}(X^\mathsf{T}Y)_{i,:} & \text{else .} \end{cases} \tag{8}$$

- *Using the* $\ell_{2,1}$*-norm, the optimal solution of* (7) *is given by the matrix* $\widehat{H} \in \mathbb{R}^{N\times n}$*, where for* $1 \leq i \leq N$,

$$\widehat{H}_{i,:} = \frac{1}{1+\alpha\lambda_i}\Big(1 - \frac{\beta}{2}\frac{1}{\|(X^\mathsf{T}Y)_{i,:}\|_2}\Big)_+ (X^\mathsf{T}Y)_{i,:} \ , \tag{9}$$

  where $(t)_+ \triangleq \max\{0, t\}$ *is the positive part function.*

3.2 Optimization with respect to Λ

For fixed H and U, the optimization Problem (5) with respect to Λ is:

$$\min_{\Lambda} \ \alpha \underbrace{\operatorname{tr}(HH^\mathsf{T}\Lambda)}_{\|\Lambda^{1/2}H\|_F^2} + \frac{1}{t}\phi(U,\Lambda) \quad \text{s.t.} \quad \begin{cases} \Lambda = \operatorname{diag}(0,\lambda_2,...,\lambda_N) \succeq 0 \ , & \text{(c)} \\ \operatorname{tr}(\Lambda) = N \in \mathbb{R}_*^+ \ . & \text{(d)} \end{cases} \tag{10}$$

This objective function is differentiable and convex with respect to Λ, and the constraints define a Simplex. Thus, several convex optimization solvers can be employed, such as those implemented in `CVXPY` (Diamond and Boyd, 2016). Popular algorithms are interior-point methods or projected gradient descent methods (Maingé, 2008). Using one algorithm of the latter type, we compute the gradient of (10) and project each iteration onto the Simplex (Duchi et al., 2008).

3.3 Optimization with respect to U

For fixed H and Λ, the optimization Problem (5) with respect to U is:

$$\min_{U} \left\| Y - X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U \end{bmatrix} H \right\|_F^2 + \frac{1}{t}\phi(U,\Lambda) \qquad \text{s.t.} \qquad U^\mathsf{T}U = I_{(N-1)} \ . \quad \text{(a')} \tag{11}$$

The objective function is not convex but twice differentiable and the constraint (a') involves the set of orthogonal matrices $Orth(N-1)$ which is not convex. Orthogonality constraint

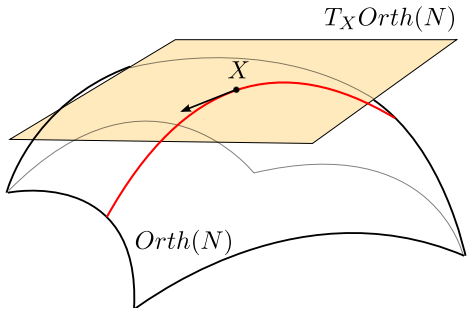


Figure 2: The principle of the manifold gradient descend given schematically. $T_X Orth(N)$ is the tangent space of $Orth(N)$ at X. The red line corresponds to a curve in $Orth(N)$ passing through the point X in the direction of the arrow. At each iteration, considering that X is the point of the current solution, a search direction belonging to $T_X Orth(N)$ is first defined, and then a descent along a curve of the manifold is performed (at the direction of the black arrow along the red line).

is central to many machine learning optimization problems including Principal Component Analysis (PCA), Sparse PCA, and Independent Component Analysis (ICA) (Hyvärinen and Oja, 2000; Zou et al., 2006; Shalit and Chechik, 2014). Unfortunately, optimizing over this constraint is a major challenge since simple updates such as matrix addition usually break orthonormality. One class of algorithms tackles this issue by taking into account that the orthogonal group $Orth(N)$ is a Riemannian submanifold embedded in $\mathbb{R}^{N\times N}$. In this article, we focus on manifold adaptation of descent algorithms to solve Problem (11).

The generalization of gradient descent methods to a manifold consists in selecting, at each iteration, a search direction belonging to the tangent space of the manifold defined at the current point X, and then performing a descent along a curve of the manifold. Figure 2 pictures this principle.

Definition 9 (Tangent space at a point of $Orth(N)$) – *Let $X \in Orth(N)$. The tangent space of $Orth(N)$ at point X, denoted by $T_X Orth(N)$ is a $\frac{1}{2}N(N-1)$ dimensional vector space defined by:*

$$T_X Orth(N) = \left\{ X\Omega \quad | \quad \Omega \in \mathbb{R}^{N\times N} \textit{is skew-symmetric} \right\}.$$

When we endow each tangent space with the standard inner product, we are able to define a notion of Riemannian gradient that allows us to find the best direction for the descent. For an objective function $\bar{f} : \mathbb{R}^{N\times N} \to \mathbb{R}$, the Riemannian gradient defined over $Orth(N)$ is given by:

$$\text{grad}\bar{f}(X) = P_X(\nabla_X \bar{f}(X)) \,, \tag{12}$$

where P_X is the projection onto the tangent space at X, which is equal to $P_X(\xi) = \frac{1}{2}X(X^\mathsf{T}\xi - \xi^\mathsf{T}X)$, and ∇_X is the standard Euclidean gradient. At each iteration, the manifold gradient descent computes the Riemannian gradient (12) that gives a direction in the tangent space. Then the update is given by applying a *retraction* onto this direction, up to a step-size. A retraction consists in an update mapping from the tangent space to the manifold, and there are many possible ways to perform that (Edelman et al., 1998; Absil et al., 2009; Arora, 2009; Meyer, 2011). From the last equation, we see that in order to solve

Problem (11) with this method, we need the Euclidean gradient of the objective function, namely those of $f(\cdot)$ and $\phi(\cdot)$. These are given in the following proposition.

Proposition 10 (Euclidean gradient with respect to U) – *The Euclidean gradient of $f(\cdot)$ and $\phi(\cdot)$ with respect to U are:*

$$\nabla_U f(H, U, \Lambda) = -2\big[(HY^\mathsf{T}X_0)_{2:,2:}\big]^\mathsf{T} + 2U(HH^\mathsf{T})_{2:,2:} \,,$$

$$\nabla_U \phi(U, \Lambda) = -\sum_{k=1}^{N-1}\sum_{\ell>k}^{N} \frac{\Big(B_{k,\ell} + B_{k,\ell}^\mathsf{T}\Big)U\Lambda_{2:,2:}}{h(U,\Lambda)_{k,\ell}} \,,$$

with $\forall 1 \leq k, \ell \leq N, B_{k,\ell} = \big(X_0^\mathsf{T} e_k e_\ell^\mathsf{T} X_0\big)_{2:,2:}$, and $h(\cdot)$ from Definition 7.

3.4 Log-barrier method and initialization

Choice of the t parameter. The quality of the approximation of Problem (4) by Problem (5) improves as $t > 0$ grows. However, taking a too large t at the beginning may lead to numerical issues. As a solution, we use the path-following method, which computes the solution for a sequence of increasing values of t until the desired accuracy. This method requires an initial value for t, denoted $t^{(0)}$, and a parameter μ such that $t^{(\ell+1)} = \mu t^{(\ell)}$. For an in-depth discussion we refer to Boyd and Vandenberghe (2004).

Initialization. At the beginning, our IGL-3SR method requires a feasible solution to initialize the algorithm. One possible choice is to take U as the identity matrix I_{N-1} and to replace (X_0, Λ) by the eigenvalue decomposition of the complete graph with trace equals to N. Indeed, its eigenvalue decomposition will always satisfy the constraints and belong to the domain of the barrier function. The initialization of H is not needed as we start directly with the H-step.

IGL-3SR is summarized in Algorithm 1.

3.5 Theoretical analysis of IGL-3SR

3.5.1 Computational complexity of IGL-3SR

Considering a graph with N nodes and $n > N$ graph signals:

- H-step (non-iterative) – The closed-form solution requires to compute the matrix product $X^\mathsf{T}Y$, which is of complexity $\mathcal{O}(nN^2)$.
- Λ-step (iterative) – When using a projected gradient descent method, the complexity of each iteration is $\mathcal{O}(nN^2)$ to compute the gradient and $\mathcal{O}(N\log(N))$ for the projection (Duchi et al., 2008). Hence, denoting by τ_Λ the number of iterations in each Λ-step, the complexity is $\mathcal{O}(\tau_\Lambda \cdot nN^2)$.
- X-step (iterative) – The complexity of each iteration is $\mathcal{O}(nN^2)$ to compute the Riemannian gradient and $\mathcal{O}(N^3)$ when we use the QR factorization as retraction (Boyd and Vandenberghe, 2018). Hence, denoting by τ_X the number of iterations in each X-step, the complexity is $\mathcal{O}(\tau_X \cdot nN^2)$.

Algorithm 1 The IGL-3SR algorithm with $\ell_{2,1}$-norm

Input: $Y \in \mathbb{R}^{N \times n}, \alpha, \beta$
Input of the barrier method: $t^{(0)}, t_{\max}, \mu$ – see Section 3.4
Output: $\widehat{H}$, $\widehat{X}$, $\widehat{\Lambda}$
Initialization: L_0 (e.g. with a complete graph) – see Section 3.4

$t \leftarrow t^{(0)}$
$(X_0, \Lambda) \leftarrow \text{SVD}(L_0)$
$U \leftarrow I_{N-1}$
while $t \leq t_{\max}$ **do**
  while not convergence **do**

▷ H-step: Compute the closed-form solution of Proposition (8)

for $1 = 1, ..., N$ **do**

$$H_{i,:} \leftarrow \frac{1}{1+\alpha\lambda_i}\Big(1 - \frac{\beta}{2}\frac{1}{\|(X^\mathsf{T}Y)_{i,:}\|_2}\Big)_+ (X^\mathsf{T}Y)_{i,:}$$

end for

▷ Λ-step: Solve Problem (10)

$$\Lambda \leftarrow \arg\min_{\Lambda} \alpha \operatorname{tr}(HH^\mathsf{T}\Lambda) + \frac{1}{t}\phi(U,\Lambda) \quad \text{s.t.} \quad \begin{cases} \Lambda = \operatorname{diag}(0, \lambda_2, ..., \lambda_N) \succeq 0\,, \\ \operatorname{tr}(\Lambda) = N \in \mathbb{R}_*^+ \end{cases}$$

▷ U-step: Solve Problem (11)

while not convergence **do**

$$U \leftarrow retraction(U(\big[(HY^\mathsf{T}X_0)_{2:,2:}\big]U - U^\mathsf{T}\big[(HY^\mathsf{T}X_0)_{2:,2:}\big]^\mathsf{T}))$$

end while

  end while
  $t \leftarrow \mu t$
end while

Overall – The complexity to go through the big loop of IGL-3SR once (i.e. once through each of the H, Λ, and X steps) is of order $\mathcal{O}(\max(\tau_\Lambda, \tau_X) \cdot nN^2)$. However, recall that τ_Λ and τ_X can be large in practice for reaching a good solution. In the following, we propose a relaxation for a faster resolution that relies on closed-form solutions.

3.5.2 Convergence properties of IGL-3SR

We now present a convergence result for IGL-3SR (Algorithm 1). Although the problem 5 that is solved is not convex, the following Lemma gives necessary conditions for a solution to be optimal.

Lemma 11 (Optimality conditions) – *Karush-Khun-Tucker (KKT) conditions are necessary conditions for any optimal solution* (H^*, U^*, Λ^*) *of Problem 5.*

The proof is given in Section 9 and relies on the fact that the orthogonality constraint verify the Linear Independence Constraint Qualification (LICQ).

The following result states that IGL-RSR converges and that any limit point of the iterates belongs to the set of KKT solutions.

Theorem 12 (Convergence of IGL-3SR) – *The sequence* $\{(H^{(\ell)}, U^{(\ell)}, \lambda^{(\ell)})\}_{\ell \geq 0}$ *generated by Algorithm 1 admits at least one converging subsequence. If we also assume that each sub-problem is effectively minimized at each iteration, then, any accumulation point of the sequence satisfies the KKT conditions of Problem 5.*

4. A relaxation for a faster resolution: the FGL-3SR algorithm

In this section, we propose another algorithm, first introduced in Le Bars et al. (2019), called *Fast Graph Learning for Smooth and Sparse Spectral Representation* (FGL-3SR) to approximately solve the initial Problem (3). FGL-3SR has a significantly reduced computational complexity due to a well-chosen relaxation. As in the previous section, we use a block-coordinate descent on H, X, and Λ, which permits to decompose the problem in three partial minimizations. FGL-3SR relies on a simplification of the minimization step in X by removing the constraint (3b). This simplification allows us to compute a closed-form on this step which greatly accelerates the minimization. However, the constraints (3a) and (3b) are equally important to obtain a valid Laplacian matrix at the end, and reducing the problem does not ensure that the constraint (3b) will be satisfied. The following proposition explains why we can get rid of constraint (3b) at the X-step, while still being able to ensure that the matrix will be a proper Laplacian at the end of the algorithm.

Proposition 13 (Feasible eigenvalues) – *Given any* $X \in \mathbb{R}^{N \times N}$ *being an orthogonal matrix with first column being equal to* $\frac{1}{\sqrt{N}}\mathbf{1}_N$ *(constraint* (3a)*), there always exists a matrix* $\Lambda \in \mathbb{R}^{N \times N}$ *such that the following constraints are satisfied:*

$$\begin{cases} (X\Lambda X^\top)_{i,j} \leq 0 \quad i \neq j \ , & \text{(3b)} \\ \Lambda = \operatorname{diag}(0, \lambda_2, ..., \lambda_N) \succeq 0 \ , & \text{(3c)} \\ \operatorname{tr}(\Lambda) = c \in \mathbb{R}_*^+ \ . & \text{(3d)} \end{cases}$$

In Proposition 14 of the next section, we will see that, by ignoring constraint (3b) at the X-step, we can compute a closed-form solution to the optimization problem. For this reason, we propose to use the closed-form solution that we derive to learn X, and right after always optimize with respect to Λ. Hence, we are sure that we will obtain a proper Laplacian at the end of the process (Proposition 13). The initialization and the optimization with respect to H are not concerned by this relaxation and can therefore be performed as in IGL-3SR (see Sections 3.1 and 3.4).

4.1 Optimization with respect to X

As already explained, during the X-step, we solve the program

$$\min_X \|Y - XH\|_F^2 \qquad \text{s.t.} \quad X^\top X = I_N, \ x_1 = \frac{1}{\sqrt{N}}\mathbf{1}_N \ , \quad \text{(3a)} \tag{13}$$

where the constraint (3b) is missing. The closed-form solution is given next.

Proposition 14 (Closed-form solution of Problem (13)) – *Let* X_0 *be any matrix that belongs to the constraints set (3a), and* $M = (X_0^\top Y H^\top)_{2:,2:}$ *the submatrix containing everything but the input's first row and first column. Finally, let* $PDQ^\top$ *be the SVD of* M*. Then,*

Algorithm 2 The FGL-3SR algorithm with $\ell_{2,1}$-norm

Input : $Y \in \mathbb{R}^{N \times n}$, α, β
Output : $\widehat{H}$, $\widehat{X}$, $\widehat{\Lambda}$
Initialization: L_0 (e.g. with a complete graph) – see Section 3.4

$(X, \Lambda) \leftarrow \mathrm{SVD}(L_0)$
for $t = 1, 2, ...$ **do**

▷ H-step: Compute the closed-form solution of Proposition (8)
for $1 = 1, ..., N$ **do**
$$H_{i,:} \leftarrow \frac{1}{1+\alpha\lambda_i}\Big(1 - \frac{\beta}{2}\frac{1}{\|(X^\mathsf{T}Y)_{i,:}\|_2}\Big)_+ (X^\mathsf{T}Y)_{i,:}$$
end for

▷ X-step: Compute the closed-form solution of Proposition (14)
$M \leftarrow (X^\mathsf{T}YH^\mathsf{T})_{2:,2:}$
$(P, D, Q^\mathsf{T}) \leftarrow \mathrm{SVD}(M)$
$$X \leftarrow X \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & PQ^\mathsf{T} \end{bmatrix}$$

▷ Λ-step: Solve the linear Program (15)
$$\Lambda \leftarrow \arg\min_{\Lambda} \alpha\,\mathrm{tr}(HH^\mathsf{T}\Lambda) \qquad \text{s.t.} \quad \begin{cases} (X\Lambda X^\mathsf{T})_{i,j} \leq 0 \quad i \neq j \\ \Lambda = \mathrm{diag}(0, \lambda_2, ..., \lambda_N) \succeq 0 \\ \mathrm{tr}(\Lambda) = N \in \mathbb{R}_*^+ \end{cases}$$

end for

the problem admits the following closed form solution:

$$\widehat{X} = X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & PQ^\mathsf{T} \end{bmatrix}. \tag{14}$$

In practice, X_0 can be fixed to the current value of X.

4.2 Optimization with respect to Λ

With respect to Λ, the optimization Problem (3) becomes:

$$\min_{\Lambda} \alpha \underbrace{\mathrm{tr}(HH^\mathsf{T}\Lambda)}_{\|\Lambda^{1/2}H\|_F^2} \qquad \text{s.t.} \quad \begin{cases} (X\Lambda X^\mathsf{T})_{i,j} \leq 0 \quad i \neq j\,, & \text{(b)} \\ \Lambda = \mathrm{diag}(0, \lambda_2, ..., \lambda_N) \succeq 0\,, & \text{(c)} \\ \mathrm{tr}(\Lambda) = N \in \mathbb{R}_*^+\,, & \text{(d)} \end{cases} \tag{15}$$

FGL-3SR is summarized in Algorithm 2.

4.3 Computational complexity of FGL-3SR

Considering a graph with N nodes and n graph signals:

- H**-step** – The closed-form solution requires to compute the matrix product $X^\mathsf{T}Y$, which is of complexity $\mathcal{O}(nN^2)$.

- **X-step** – The closed-form solution requires to compute the SVD of $(X_0^\mathsf{T} Y H^\mathsf{T})_{2:,2:} \in \mathbb{R}^{(N-1)\times(N-1)}$, which is of complexity $\mathcal{O}(N^3)$ (Cline and Dhillon, 2006).

- **Λ-step** – Solving the LP can be done with interior-point methods or with the ellipsoid method (Vandenberghe, 2010). For accuracy ε, the ellipsoid method yields a complexity of $\mathcal{O}(\max(m, N) \cdot N^3 \log(1/\varepsilon))$, where $m = \frac{1}{2}N(N-1) + N + 1$ is the number of constraints (Bubeck, 2015).

Overall – As $m > N$, the complexity for FGL-3SR is of order $\mathcal{O}(N^5)$ when using the ellipsoid method. In contrast, the most competitive related algorithm of the literature (ESA-GL (Sardellitti et al., 2019)) relies on a semi-definite program and is of order at least $\mathcal{O}(N^8)$ (see Section 6). As will be clearly demonstrated in Section 7, in practice the empirical execution time of FGL-3SR is lower than IGL-3SR and ESA-GL.

4.4 Differences between IGL-3SR and FGL-3SR

The two proposed algorithms are based on a modification of the initial optimization Problem (3). Indeed, both of them relax the constraint (3b), $\forall k \neq \ell,\ (X\Lambda X^\mathsf{T})_{k,\ell} \leq 0$, but using two different approaches. IGL-3SR approximates the initial optimization problem through the use of a log-barrier function. The advantage of the barrier is twofold: first, it allows to overcome the technical constraint (3b) and solve the program using a block-coordinate descent algorithm; second, the use of the barrier makes the block-variables separable over the constraint set, allowing the convergence of the objective function of IGL-3SR. In addition, IGL-3SR always keeps the set of variables in the initial set of constraints, essential for the matrix $X\Lambda X^\mathsf{T}$ to be a proper Laplacian.

On the other hand, FGL-3SR, instead of using a log-barrier function to relax the constraint (3*b*), it removes it from the X-step. Recall that we are perfectly able to do that as we know from Lemma 13 that for any X returned by the X-step (4.1), there exist a Λ making $X\Lambda X^\mathsf{T}$ a Laplacian. This relaxation speeds-up drastically the X-step while loosing the convergence property and the decreasing over the initial constraints set.

5. A probabilistic interpretation for the optimization problem

In this section, we introduce a new representation model adapted to smooth graph signals with sparse spectral representation. The goal of this model is to provide a probabilistic interpretation of Problem (3) and link its objective function to a maximum a posteriori estimation (Proposition 18).

Given a Laplacian matrix $L = X\Lambda X^\mathsf{T}$, we propose the following *Factor Analysis Framework* to model a graph signal y:

$$y = Xh + m_y + \varepsilon\ , \tag{16}$$

where $m_y \in \mathbb{R}^N$ is the mean of the graph signal y and ε is a Gaussian noise with zero mean and covariance $\sigma^2 I_N$. Here, the latent variable $h = (h_1, ..., h_N)$ controls y through the eigenvector matrix X of L. The choice of the representation matrix X is particularly adapted since it reflects the topology of the graph and provides a spectral embedding of its vertices. Moreover, as seen in Section 2, X can be interpreted as a graph Fourier basis,

which makes it an intuitive choice for the representation matrix. In a noiseless scenario with $m_y = 0$, h actually corresponds to the GFT of y.

To comply with the spectral sparsity assumption (Assumption 2), we now propose a distribution that allows h to admit zero-valued components. To this end, we introduce independent latent Bernoulli variables γ_i with success probability $p_i \in [0, 1]$. Knowing $\gamma_1, ..., \gamma_N$, the conditional distribution for h is:

$$h|\gamma \sim \mathcal{N}(0, \widetilde{\Lambda}^\dagger) \ , \tag{17}$$

where $\widetilde{\Lambda}^\dagger$ is the Moore-Penrose pseudo-inverse of the diagonal matrix containing the values $\{\lambda_i \mathbb{1}\{\gamma_i = 1\}\}_{i=1}^N$. In this model, γ_i controls the sparsity of the i-th element of h. Indeed, if $\gamma_i = 0$, then $h_i = 0$ almost surely. In the other hand, if $\gamma_i = 1$ then h_i follows a Gaussian distribution with zero-mean and variance equal to $1/\lambda_i$. This is adapted to the smoothness hypothesis as for high value of λ_i (high frequency), the distribution of h_i concentrates more around 0, leading to small value of $\lambda_i h_i^2$. The associated probability of success p_i can be chosen *a priori*. One way to chose it is to take p_i inversely proportional to λ_i. Indeed, this would increase the probability to be sparse at dimensions where the associated eigenvalue is high. Note that, since $\lambda_1 = 0$, h_1 follows a centered degenerate Gaussian, i.e h_1 is equal to 0 almost surely. Furthermore, if $p_i = 1$ for all i, our model reduces to the one proposed by Dong et al. (2016), which was only focused on the smoothness assumption.

Definition 15 (Prior and conditional distributions) – *The following equations summarize the prior and important conditional distributions of our model:*

$$p(h_i|\gamma_i, \lambda_i) \propto \exp(-\lambda_i h_i^2)\mathbb{1}\{\gamma_i = 1\} + \mathbb{1}\{h_i = 0, \gamma_i = 0\} \ , \tag{18}$$

$$p(y|h, X) \propto \exp(-\frac{1}{\sigma^2}\|y - Xh - m_y\|_2^2) \ , \tag{19}$$

$$p(\gamma_i) \propto p_i^{\gamma_i}(1 - p_i)^{1-\gamma_i} \ . \tag{20}$$

For simplicity, in the following we consider that $m_y = 0$ and $p_1 = 0$.

Lemma 16 – *Assume the proposed Model (16). If $p_1 = 0$ and $p_i \in (0, 1)$, $\forall i \geq 2$, then:*

$$\begin{aligned} -\log(p(h|y, X, \Lambda)) \ \propto \ & \frac{1}{\sigma^2}\|y - Xh\|_2^2 + \frac{1}{2}h^\mathsf{T}\Lambda h \\ & + \sum_{i=1}^N \mathbb{1}\{h_i \neq 0\}\left(p_i \log(\frac{\lambda_i}{\sqrt{2\pi}}) - \log(p_i) - \log(\frac{\lambda_i}{\sqrt{2\pi}})\right) . \end{aligned}$$

Definition 17 (Lambert W-Function) – *The Lambert W-Function, denoted by $W(\cdot)$, is the inverse function of $f : W \mapsto We^W$. In particular, we consider W to be the principal branch of the Lambert function, defined over $[-1/e, \infty)$.*

Proposition 18 (A posteriori distribution of h) – *Let $C > 0$, and assume for all $i \geq 2$ that $p_i = e^{-C}$ if $\lambda_i = \sqrt{2\pi}$, whereas $p_i = -W\left(-\frac{e^{-C}\log(\lambda_i/\sqrt{2\pi})}{\lambda_i/\sqrt{2\pi}}\right)\frac{1}{\log(\lambda_i/\sqrt{2\pi})}$ otherwise. Then, $p_i \in (0, 1)$ and there exist constants $\alpha, \beta > 0$ such that:*

$$-\log(p(h|y, X, \Lambda)) \ \propto \ \|y - Xh\|_2^2 + \alpha h^\mathsf{T}\Lambda h + \beta\|h\|_0 \ .$$

This proposition tells us that: *for a given Laplacian matrix, the maximum a posteriori estimate of h would corresponds to the minimum of Problem* (3).

6. Related work on GSP-based graph learning methods

Here we detail the two state-of-the-art methods for graph learning in the GSP context that are closer to our work and that will be used for our experimental comparison in Section 7.

GL-SigRep (Dong et al., 2016). This method supposes that the observed graph signals are smooth with respect to the underlying graph, but do not consider the spectral sparsity assumption. To learn the graph, they propose to solve the optimization problem:

$$\min_{L,\tilde{Y}} \|Y-\tilde{Y}\|_F^2 + \alpha\|L^{1/2}\tilde{Y}\|_F^2 + \beta\|L\|_F^2 \quad \text{s.t.} \quad \begin{cases} L_{k,\ell} = L_{\ell,k} \leq 0 \quad k \neq \ell \,, \\ L\mathbf{1} = \mathbf{0} \,, \\ \operatorname{tr}(L) = N \in \mathbb{R}_*^+ \,. \end{cases} \tag{21}$$

Remark that since no constraints are imposed on the spectral representation of the signals, the Laplacian matrix is directly learned. The optimization procedure to solve (21) consists in an alternating minimization over L and $\tilde{Y}$. With respect to $\tilde{Y}$ the problem has a closed-form solution whereas for L, the authors propose to use a Quadratic Program solver involving $\frac{1}{2}N(N-1)$ parameters and $\frac{1}{2}N(N-1)+N+1$ constraints.

ESA-GL (Sardellitti et al., 2019). This is a two-step algorithm where the signals are supposed to admit a sparse representation with respect to the learned graph. The difference to our paper is two-fold. First, ESA-GL does not include the smoothness assumption while learning the Fourier basis X. This brings a different two-step optimization program. Second, the complexity of the ESA-GL algorithm (at least $\mathcal{O}(N^8)$) is much higher than ours ($\mathcal{O}(N^5)$ for FGL-3SR - see Section 4.3), and hence is prohibitive for large graphs. The first step consists in fitting an orthonormal basis such that the observed graph signals Y admit a sparse representation with respect to this basis. They consider the problem:

$$\min_{H,X} \|Y-XH\|_F^2 \qquad \text{s.t.} \quad \begin{cases} X^\intercal X = I_N, \; x_1 = \frac{1}{\sqrt{N}}\mathbf{1}_N \,, \\ \|H\|_{2,0} \leq K \in \mathbb{N} \,, \end{cases} \tag{22}$$

which is solved using an alternating minimization. Once estimates for H and X have been computed, they solve a second optimization problem in order to learn the Laplacian L associated to the learned basis $\widehat{X}$. This is done by minimizing:

$$\min_{L\in\mathbb{R}^{N\times N}, C_K\in\mathbb{R}^{K\times K}} \operatorname{tr}(\widehat{H}_K^T C_K \widehat{H}_K) + \mu\|L\|_F^2 \quad \text{s.t.} \quad \begin{cases} L_{k,\ell} = L_{\ell,k} \leq 0 \quad k \neq \ell \,, \\ L\mathbf{1}_N = \mathbf{0}_N \,, \\ L\widehat{X}_K = \widehat{X}_K C_K, \quad C_K \succeq 0 \,, \\ \operatorname{tr}(L) = N \in \mathbb{R}_*^+ \,, \end{cases} \tag{23}$$

where $C_K \in \mathbb{R}^{K\times K}$ and $\widehat{X}_K$ corresponds to the columns of $\widehat{X}$ associated to the non-zero rows of $\widehat{H}$ denoted $\widehat{H}_K$. Thus, the second step aims at estimating a Laplacian that enforces the smoothness of the learned signal representation $\widehat{X}\widehat{H}$. This semi-definite program requires the computation of over $\frac{1}{2}N(N-1)+\frac{1}{2}K(K-1)$ parameters that, as we show empirically in the next section, can be difficult to compute for graphs with large number of nodes. For more details on the optimization program and the additional matrix C_K, the readers shall refer to the aforementioned paper.

7. Experimental evaluation

The two proposed algorithms, IGL-3SR and FGL-3SR, are now evaluated and compared with the two state-of-the-art methods presented earlier, GL-SigRep and ESA-GL. The results of our empirical evaluation are organized in three subsections: Section 7.2 and 7.3 use synthetic data for first comparing the different methods and then for studying the influence of the hyperparameters; Section 7.4 displays several examples on real-world data.

All experiments were conducted on a single personal computer: a personal laptop with with 4-core 2.5GHz Intel CPUs and Linux/Ubuntu OS. For the Λ-step of both algorithms, we use the Python's `CVXPY` package (Diamond and Boyd, 2016). For the X-step of IGL-3SR, we use the conjugate gradient descent solver combined with an adaptive line search, both provided by `Pymanopt` (Townsend et al., 2016), a Python toolbox for optimization on manifolds. Note that this package only requires the gradients, which in this case are given in Proposition 10. Graphs are generated with `NetworkX` (Hagberg et al., 2008), `NetworKit` (Staudt et al., 2016), and `SNAP` (Leskovec and Sosič, 2016). The source code of our implementations is available at `https://github.com/pierreHmbt/GL-3SR`.

7.1 Evaluation metrics

We provide visual and quantitative comparisons of the learned Laplacian $\widehat{L}$ and its weight matrix $\widehat{W}$ using the performance measures: *Recall*, *Precision*, and F_1-*measure*, which are standard for this type of evaluation (Pasdeloup et al., 2017). The F_1-measure evaluates the quality of the estimated support – the non-zero entries – of the graph and is given by:

$$F_1 = \frac{2 \times precision \times recall}{precision + recall} \ .$$

As in Pasdeloup et al. (2017), the F_1-measure is computed on a thresholded version of the estimated weight matrix $\widehat{W}$. This threshold is equal to the average value of the off-diagonal entries of $\widehat{W}$ (same process as in (Sardellitti et al., 2019)).

In addition, we compute the correlation coefficient $\rho(L, \widehat{L})$ between the true Laplacian entries $L_{i,j}$ and their estimates $\widehat{L}_{i,j}$:

$$\rho(L, \widehat{L}) = \frac{\sum_{ij}(L_{ij} - L_m)(\widehat{L}_{ij} - \widehat{L}_m)}{\sqrt{\sum_{ij}(L_{ij} - L_m)^2}\sqrt{\sum_{ij}(\widehat{L}_{ij} - \widehat{L}_m)^2}} \ , \tag{24}$$

where L_m and $\widehat{L}_m$ are the average values of the entries of the true and estimated Laplacian matrices, respectively. This ρ coefficient evaluates the quality of the weights distribution over the edges.

7.2 Experiments on synthetic data

We now evaluate and compare all algorithms on several types of synthetic data. Details about graphs, associated graph signals, and evaluation protocol used for the experiments, are detailed in the sequel. Regarding the the computational times reported, these were measured on a personal laptop with 4-core 2.5GHz Intel CPUs and Linux/Ubuntu OS.

N	Metrics	Random Geometric graphs (RG) IGL-3SR	FGL-3SR	ESA-GL	GL-SigRep	Erdős-Rényi graphs (ER) IGL-3SR	FGL-3SR	ESA-GL	GL-SigRep
20	Precision	**0.973** (±**0.042**)	0.952 (±0.042)	0.899 (±0.054)	0.929 (±0.068)	**0.952** (±**0.045**)	0.819 (±0.080)	0.931 (±0.045)	0.704 (±0.125)
	Recall	0.974 (±0.018)	**0.985** (±**0.023**)	0.968 (±0.052)	0.967 (±0.028)	0.927 (±0.046)	0.824 (±0.105)	**0.951** (±**0.041**)	0.899 (±0.075)
	F_1-measure	**0.974** (±**0.028**)	0.968 (±0.027)	0.929 (±0.032)	0.947 (±0.040)	0.938 (±0.028)	0.816 (±0.068)	**0.941** (±**0.038**)	0.779 (±0.071)
	$\rho(L, \overline{L})$	**0.938** (±**0.052**)	0.903 (±0.029)	0.925 (±0.050)	0.786 (±0.037)	**0.917** (±**0.035**)	0.730 (±0.063)	0.897 (±0.045)	0.599 (±0.074)
	Time	< 1min	< 10s	< **5s**	< **5s**	< 1min	< 10s	< **5s**	< **5s**
50	Precision	**0.901** (±**0.022**)	0.817 (±0.041)	0.845 (±0.088)	0.791 (±0.055)	0.820 (±0.027)	0.791 (±0.047)	**0.854** (±**0.038**)	0.476 (±0.037)
	Recall	0.902 (±0.018)	0.807 (±0.036)	**0.910** (±**0.040**)	0.720 (±0.059)	0.812 (±0.042)	0.740 (±0.049)	0.830 (±0.051)	**0.856** (±**0.023**)
	F_1-measure	**0.901** (±**0.014**)	0.812 (±0.017)	0.868 (±0.036)	0.750 (±0.001)	0.815 (±0.021)	0.761 (±0.031)	**0.841** (±**0.021**)	0.610 (±0.026)
	$\rho(L, \overline{L})$	**0.863** (±**0.020**)	0.743 (±0.031)	0.832 (±0.033)	0.549 (±0.022)	0.783 (±0.026)	0.728 (±0.020)	**0.816** (±**0.058**)	0.480 (±0.002)
	Time	< 17mins	< **40s**	< 60s	< **40s**	< 17mins	< **40s**	< 60s	< **40s**
100	Precision	**0.713** (±**0.012**)	0.711 (±0.029)	0.667 (±0.022)	–	**0.677** (±**0.044**)	0.640 (±0.033)	0.654 (±0.038)	–
	Recall	**0.751** (±**0.067**)	0.584 (±0.011)	0.743 (±0.017)	–	0.580 (±0.021)	0.543 (±0.027)	**0.637** (±**0.023**)	–
	F_1-measure	**0.732** (±**0.034**)	0.641 (±0.010)	0.703 (±0.012)	–	**0.623** (±**0.009**)	0.586 (±0.016)	0.589 (±0.019)	–
	$\rho(L, \overline{L})$	**0.612** (±**0.045**)	0.483 (±0.015)	0.596 (±0.033)	–	0.551 (±0.016)	0.512 (±0.0223)	**0.644** (±**0.023**)	–
	Time	< 50mins	< **2mins**	< 4mins	–	< 50mins	< **2mins**	< 4mins	–

Table 1: Comparison of the four methods on five quality metrics (avg ± std) for graphs of $N = \{20, 50, 100\}$ nodes, and for fixed number of $n = 1000$ graph signals.

Graphs and signals. We carried out experiments on four graphs with $20, 50$, and 100 vertices, following: i) a Random Geometric (RG) graph model with a 2-D uniform distribution for the coordinates of the nodes and a truncated Gaussian kernel of width size 0.5 for the edges, where weights smaller than 0.75 were set to 0; ii) an Erdős-Rényi (ER) model with edge probability 0.2; iii) a Forest Fire (FF) model with *forest fire forward/backward burning probability* equal to 0.3 (Leskovec et al., 2005, 2007; Leshakov, 2017); iv) a Stochastic Kronecker (SK) graph model with an *initiator graph* of size 3×3 where weights are $[0.75, 0.65, 0.55]$ (Leskovec et al., 2010). As we consider undirected graphs in this paper, we symmetrize the directed graphs returned by the SK model. In this model, the generated graphs are of size $3^3 = 27$, $3^4 = 81$, and $3^5 = 243$.

Given a graph, the sampling process was made according to Model (18) that we presented in Section 5. The mean value of each signal was set to 0, the variance of the noise was set to 0.5, and the sparsity was chosen to obtain observations with k-sparse spectral representation, where k is equal to half the number of nodes (i.e $10, 20, 50$).

For each type of graph, we ran 10 experiments with 1000 graph signals generated as explained above. For all the methods, the hyperparameters α and β are set by maximizing the F_1-measure on the thresholded $\widehat{W}$, as explained in Section 7.1.

Choice of $\|\cdot\|_S$. In the following we make all experiments for IGL-3SR and FGL-3SR with the $\ell_{2,1}$-norm. This is motivated by an important fact brought by the closed-form solutions given in Proposition 8. Indeed, for $\ell_{2,1}$-norm, the sparsity of $\widehat{H}$ is only controlled by β (Equation (9)). On the contrary, when using the $\ell_{2,0}$-norm, the value of α also influences the sparsity (Equation (8)). This is an important behavior, as the tuning of β and α becomes *independent* – at least with respect to the H-step – and therefore, as we will see in Section 7.3.1, easier to tune.

Quantitative results. Average evaluation metrics and their standard deviation are collected in Tables 1 and 2. The results show that the use of the sparsity constraint improves the quality of the learned graphs. Indeed, the two proposed methods IGL-3SR

		Forest Fire graphs (FF)					*Kronecker graphs (SK)*			
N	**Metrics**	**IGL-3SR**	**FGL-3SR**	**ESA-GL**	**GL-SigRep**	N	**IGL-3SR**	**FGL-3SR**	**ESA-GL**	**GL-SigRep**
	Precision	0.910 (±0.052)	0.870 (±0.052)	**0.935 (±0.062)**	0.898 (±0.068)		0.947 (±0.022)	0.916 (±0.040)	**0.955 (±0.031)**	0.802 (±0.115)
	Recall	**0.988 (±0.017)**	0.876 (±0.017)	0.925 (±0.042)	0.895 (±0.073)		**0.974 (±0.026)**	0.927 (±0.030)	0.963 (±0.036)	0.945 (±0.051)
20	F_1-measure	**0.947 (±0.030)**	0.872 (±0.030)	0.929 (±0.038)	0.893 (±0.046)	27	**0.960 (±0.013)**	0.921 (±0.020)	**0.959 (±0.025)**	0.862 (±0.074)
	$\rho(L, \overline{L})$	**0.915 (±0.038)**	0.796 (±0.038)	0.850 (±0.039)	0.773 (±0.040)		**0.975 (±0.006)**	0.939 (±0.013)	**0.974 (±0.010)**	0.825 (±0.067)
	Time	< 1min	< 10s	< **5s**	< **5s**		< 1min	< 10s	< **5s**	< **5s**
	Precision	0.731 (±0.072)	0.664 (±0.072)	**0.812 (±0.076)**	0.754 (±0.050)		0.894 (±0.020)	0.824 (±0.105)	**0.951 (±0.041)**	0.894 (±0.028)
	Recall	**0.896 (±0.122)**	0.745 (±0.122)	0.743 (±0.079)	0.828 (±0.039)		0.852 (±0.035)	0.819 (±0.080)	**0.931 (±0.045)**	0.810 (±0.026)
50	F_1-measure	**0.803 (±0.087)**	0.701 (±0.087)	0.774 (±0.071)	0.787 (±0.021)	81	0.872 (±0.023)	0.816 (±0.068)	**0.941 (±0.038)**	0.849 (±0.009)
	$\rho(L, \overline{L})$	**0.763 (±0.096)**	0.655 (±0.096)	0.700 (±0.078)	0.671 (±0.034)		0.868 (±0.021)	0.730 (±0.063)	**0.897 (±0.045)**	0.804 (±0.019)
	Time	< 17mins	< **40s**	< 60s	< **40s**		< 17mins	< **40s**	< 60s	< 6mins
	Precision	**0.794 (±0.108)**	0.788 (±0.096)	0.669 (±0.047)	–		**0.884 (±0.014)**	0.841 (±0.014)	–	–
	Recall	0.719 (±0.061)	0.683 (±0.066)	**0.749 (±0.037)**	–		**0.800 (±0.015)**	0.705 (±0.028)	–	–
100	F_1-measure	**0.748 (±0.039)**	0.727 (±0.034)	0.705 (±0.021)	–	243	**0.828 (±0.015)**	0.767 (±0.012)	–	–
	$\rho(L, \overline{L})$	**0.743 (±0.029)**	0.726 (±0.026)	0.689 (±0.022)	–		**0.829 (±0.011)**	0.819 (±0.013)	–	–
	Time	< 50mins	< **2mins**	< 4mins	–		< 80mins	< **3mins**	–	–

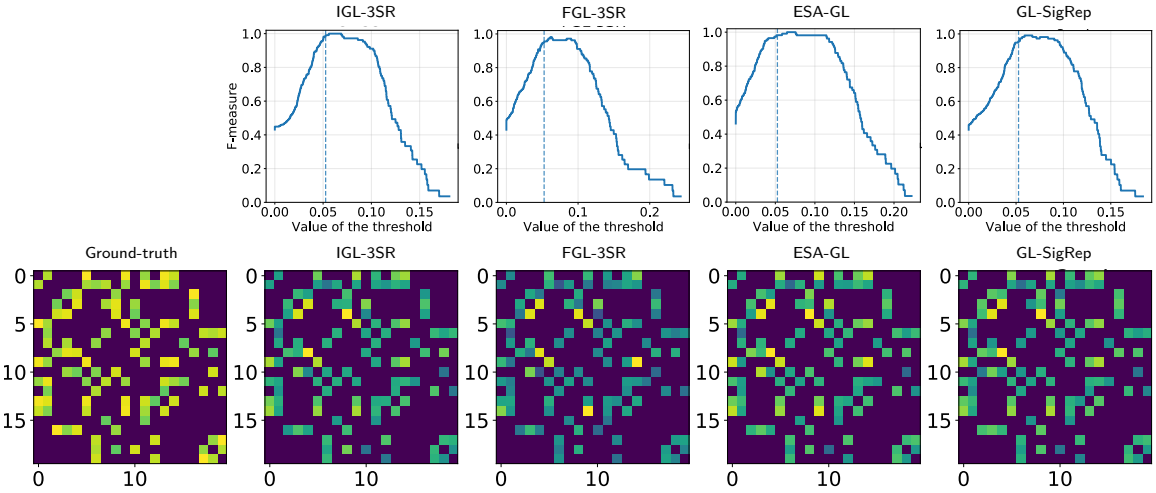
Table 2: Comparison of the four methods on five quality metrics (avg ± std) for graphs of $N = \{20, 50, 100\}$ and $N = \{27, 81, 243\}$ nodes, and for fixed number of $n = 1000$ graph signals.

and FGL-3SR, as well as ESA-GL, have better overall performance in all the metrics than GL-SigRep that only considers the smoothness aspect. This had to be expected as our methods match perfectly to the sparse (bandlimited) condition.

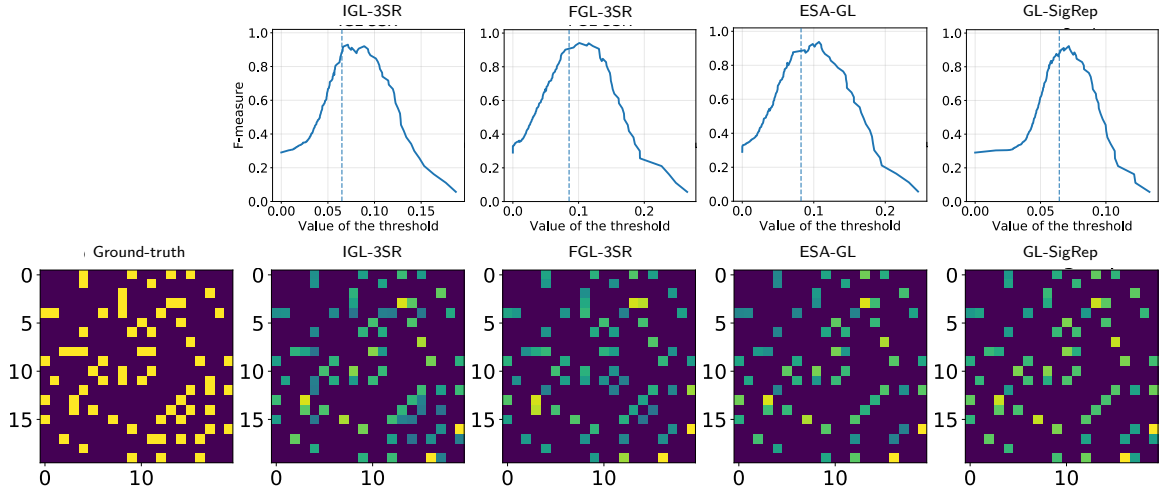
Comparing the results across the different types of synthetic graphs, our methods are robust, while being more efficient on RG and FF graphs. In general, IGL-3SR and FGL-3SR present similar performance to ESA-GL but IGL-3SR seems preferable in the case of RG and FF graphs. When more than 100, respectively 200, nodes are considered, the computational resources necessary for GL-SigRep, respectively ESA-GL, are too demanding and render the problems intractable. We can see that, while IGL-3SR has better results than FGL-3SR, the time necessary to estimate the graph is much longer. In addition, examples of learned graphs are displayed in Figure 3 with the ground-truth on the left and the learned weighted adjacency matrices (after thresholding). The evolution of the F_1-measure regarding the value of the threshold is also displayed and shows that a large range of threshold could have been used to obtain similar performance. All these results, combined with those of Tables 1 and 2, indicate that in this sampling process the proposed FGL-3SR method managed to infer accurate graphs despite the relaxation.

Speed performance. Figure 4 displays the evolution of the empirical computation time as the number of nodes increases. For each algorithm, time per iteration is: i) for IGL-3SR and FGL-3SR, the time needed for the computation of the 3 steps one time; ii) for ESA-3SR,the time needed for the computation of the quadratic program; iii) for GL-SigRep, the time needed for the computation of its two steps one time. FGL-3SR appears to be much faster than the other methods. Furthermore, we observe that our methods are scalable over a wider range of graph sizes than the competitors. Indeed, even quite small graphs of 100 and 150 nodes, respectively, were already too 'large' for the two competitors to be able to produce results, and they even led to memory allocation errors.

IGL-3SR v.s. FGL-3SR. In terms of numerical performance, IGL-3SR is better than FGL-3SR (Table 1). Indeed, except for graphs of size 20, the metrics related to the recovery



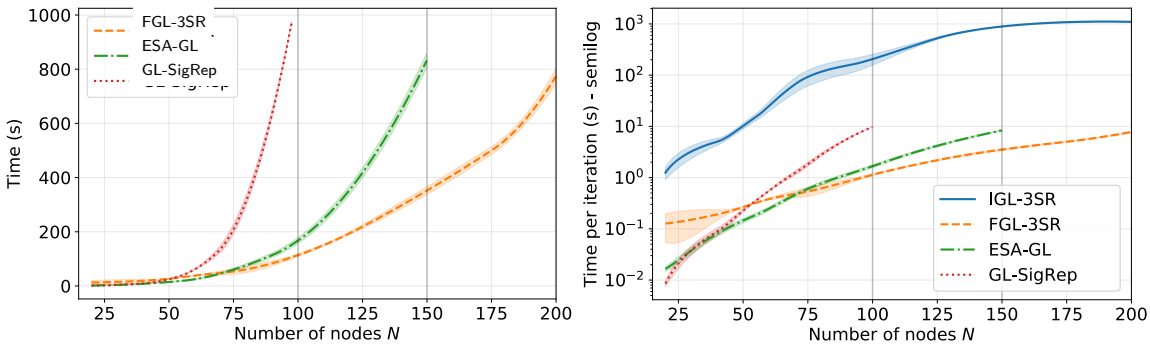
(a) Graph learning on RG synthetic graphs.



(b) Graph learning on ER synthetic graphs.

Figure 3: Graph learning results on random synthetic graphs of 20 nodes: (a) for a RG graph, and (b) for an ER graph. Each of the two subfigures presents: (top row) the evolution of the F_1-measure with respect to different threshold values and the dashed line indicates the chosen threshold value; (bottom row) shows as leftmost the ground truth adjacency matrix, followed by the respective learned adjacency matrices (thresholded) by the compared methods.

of the true graph do give better results. On the contrary, in terms of computational cost, FGL-3SR is better than IGL-3SR (see Figure 4). Indeed, no matter the size of the graph, FGL-3SR has lower time per iteration than IGL-3SR. This is due to the fact that, contrary to IGL-3SR which solves two out of three sub-problems with iterative methods, FGL-3SR solves two sub-problems via closed-form solutions that can be computed efficiently. In



(a) Standard scale. **(b)** Semi-log scale.

Figure 4: Average and standard deviation of the computation time over 10 trials for IGL-3SR, FGL-3SR, ESA-GL, and GL-SigRep, as the number of nodes increases. GL-SigRep and ESA-GL failed to produce a result for graphs with more than 100 and 150 nodes, respectively. (a) The total computation times, and (b) the time needed for a single iteration of each algorithm.

conclusion, when the number of nodes is small, IGL-3SR is preferred. If not, one should use FGL-3SR.

7.3 Influence of the hyperparameters

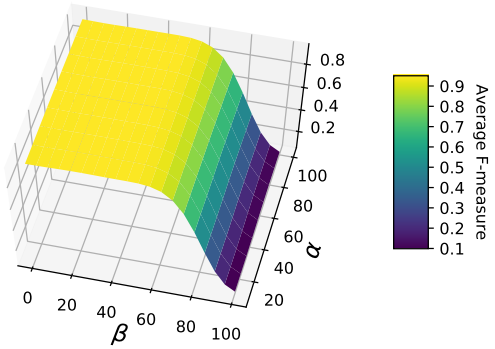
We now study how hyperparameters of IGL-3SR and FGL-3SR influence their overall performance, with respect to the F_1-measure. This study is made on a RG graph with $N = 20$ nodes and 10-bandlimited signals Y in $\mathbb{R}^{20\times 1000}$.

7.3.1 Influence of α and β

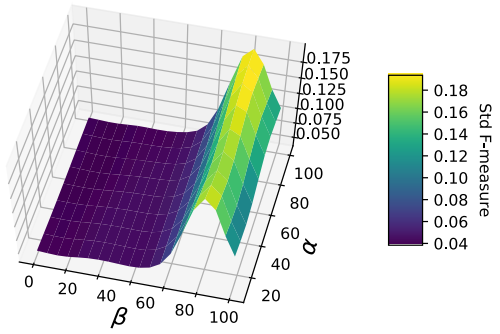
We first highlight the influence of α and β on FGL-3SR. We run and collect the F_1-measure for 20 values of α (resp. β) in $[10^{-5}, 100]$ (resp. $[10^{-5}, 60]$). The resulting heatmaps are displayed in Figure 5. The most important observation is that the value of α does not seem to impact a lot the structure of the resulted graphs. Indeed, for a fixed value of β, the F_1-measure is stable when α varies. However, it is interesting that the convergence curve of FGL-3SR (Figure 6) is directly impacted by α: large values for α tend to produce oscillations on the convergence curves. Thus, setting to a small value $\alpha > 0$ is suggested. Contrary to α, tuning the parameter β is critical since high β values cause a drastic decrease in F_1-measure. This sharp decrease appears when the chosen β imposes too much sparsity for the learned $\widehat{H}$. One may note that the best β corresponds to the value just before the sharp decrease, and this is the value that should be chosen. Although the previous analysis has been done on FGL-3SR, during our experimental studies, α and β influenced the F_1-measure similarly when using IGL-3SR.

7.3.2 Influence of t

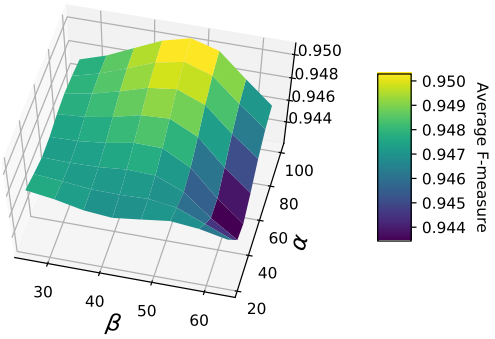
We now highlight the influence of t on IGL-3SR. Figure 7 shows the learned graphs for several values of $t \in [10, 10^4]$. This experiment brings two main messages: first, when t is too low, the learned graph is very close to the complete graph, whereas when t increases the



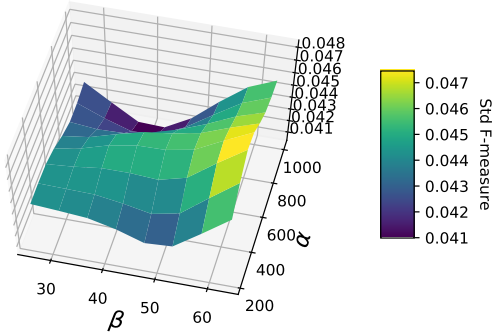
(a) Average of the F_1-measure.



(b) Standard deviation of the F_1-measure.



(c) Average of the F_1-measure (focus).



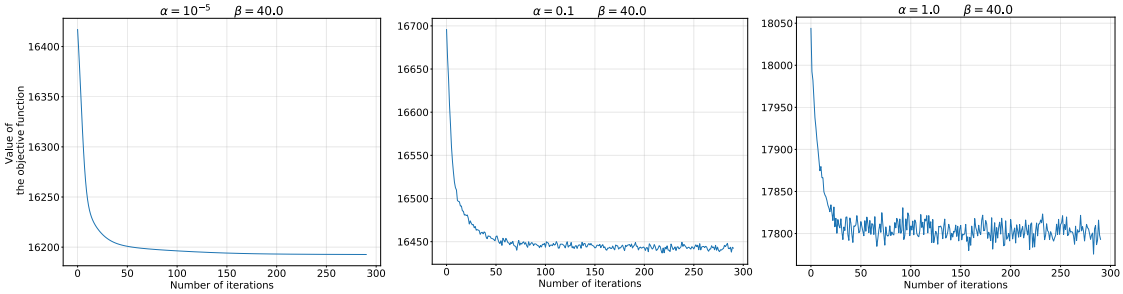
(d) Standard deviation of the F_1-measure (focus).

Figure 5: Evolution of the average (a)(c) and standard deviation (b)(d) of the F_1-measure over 10 runs of FGL-3SR on RG graphs with 20 nodes. At the top figure row $\beta \in [0, 100]$, and at the bottom row $\beta \in [20, 70]$.

learned graph becomes more structured and tends to be sparse. This result was expected since a larger t brings the barrier closer to the true constraint, i.e. we allow elements of the resulting Laplacian matrix to be closer to 0. Second, it appears that α also influences the final results in a similar way to t. Again, this was expected as the minimization of the objective function during the Λ-step of Problem (5) is equivalent to the minimization of $\mathrm{tr}(HH^\top\Lambda) + \frac{1}{\alpha t}\phi(U, \Lambda)$.

For a discussion on the initial value of t, $t^{(0)}$, and the step size μ such that $t^{(\ell+1)} = \mu t^{(\ell)}$, both relative to the barrier method, we refer the reader to (Boyd and Vandenberghe, 2004). However, recall that t is not a hyperparameter to tune in practice, and should be taken as large as possible. The mere goal is to prevent numerical issues. Fortunately, a wide range of values for $t^{(0)}$ and μ achieves that goal (Boyd and Vandenberghe, 2004).

Tuning the hyperparameters. The hyperparameter α does not seems to have a substantial impact on the F_1-measure. However, a low value of it may be preferred in FGL-3SR for convergence purpose (Figure 6). The parameter t always needs to be maximal provided that it does not cause numerical issues. Classical heuristics and methods, like the one presented in Section 3.4, can be used to tune t (Boyd and Vandenberghe, 2004). Hence,



(a) Low α value: $\alpha = 10^{-5}$. **(b)** Medium α value: $\alpha = 10^{-1}$. **(c)** High α value: $\alpha = 1$.

Figure 6: Convergence curves of the objective function as the number of iterations increases, when using FGL-3SR with different α values.

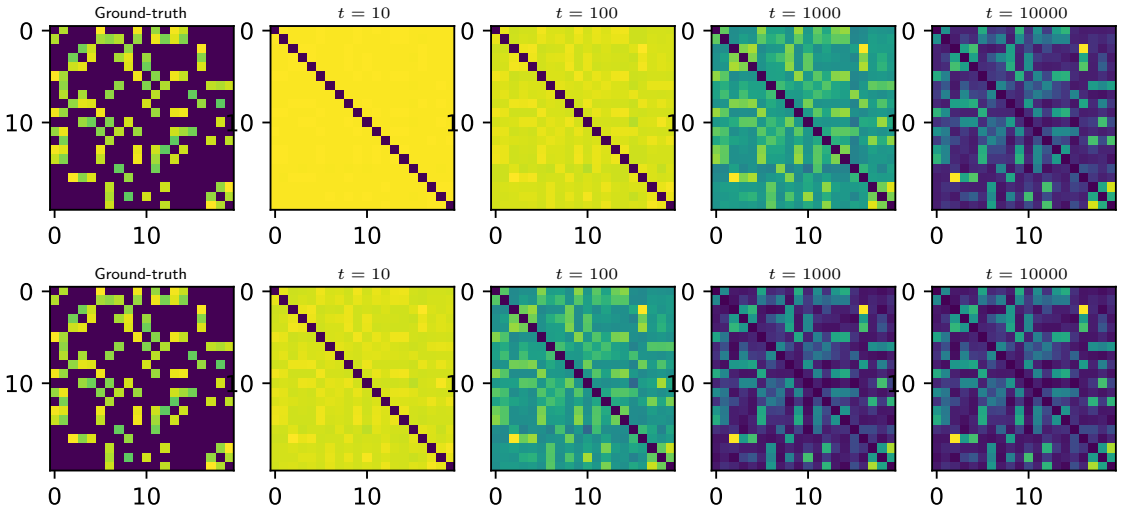
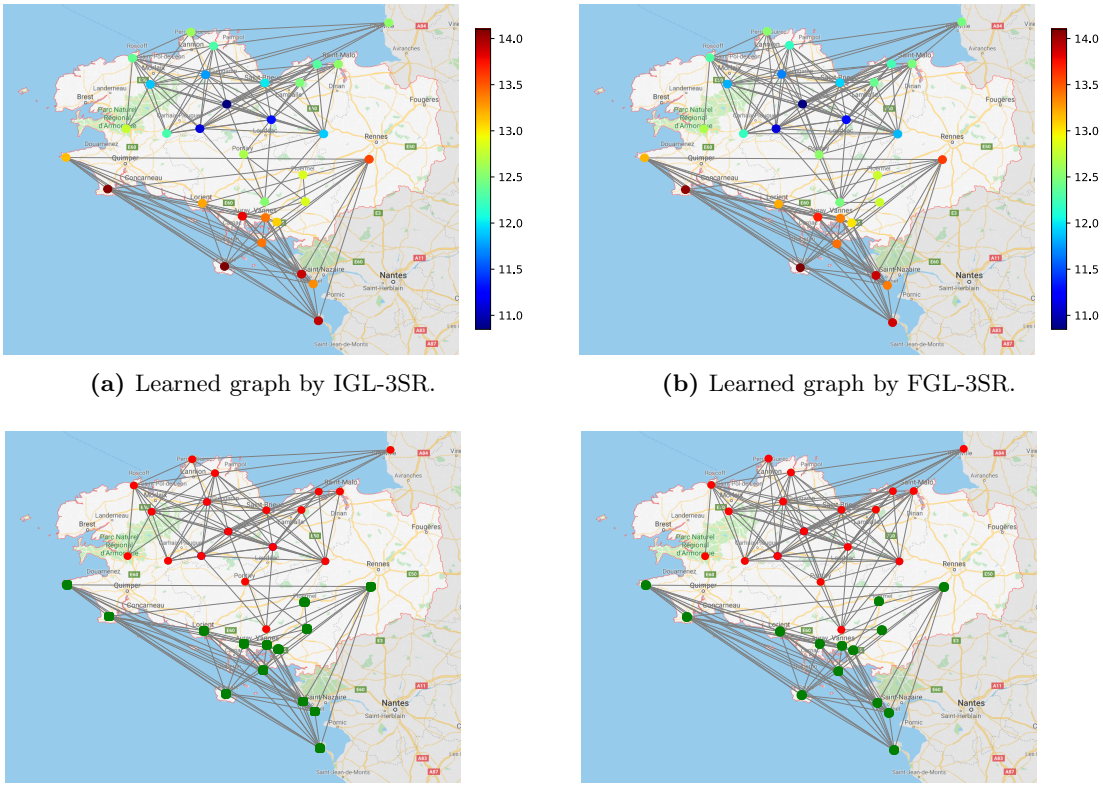


Figure 7: Learned graphs with increasing t values: (top row) $\alpha = 10^{-4}$, (bottom row) $\alpha = 10^{-3}$.

according to our experiments, it remains only β as a critical hyperparameter to tune for both these methods. Based on Figure 5, one way to fix it is to find the largest β value that leads to satisfying results in terms of signal reconstruction. Alternatively, if we have an idea about the number of clusters k that resides on the graph, we could select a β value that produces a k-sparse spectral representation. Bearing in mind that other related works require the tuning of two hyperparameters, our approach turns out to be of higher value for practical application on real data where these parameters are unknown and must be tuned.

7.4 Temperature data

We used hourly temperature (C°) measurements on 32 weather stations in Brittany, France, during a period of 31 days (Chepuri et al., 2017). The dataset contains $24 \times 31 = 744$ multivariate observations, i.e. $Y \in \mathbb{R}^{32\times 744}$, that are assumed to correspond to an unknown graph, which is our objective to infer. For our two algorithms, we set $\alpha = 10^{-4}$, and β is chosen so that we obtain a 2-sparse spectral representation, which this last assumes that there are two clusters of weather stations.



(a) Learned graph by IGL-3SR.

(b) Learned graph by FGL-3SR.

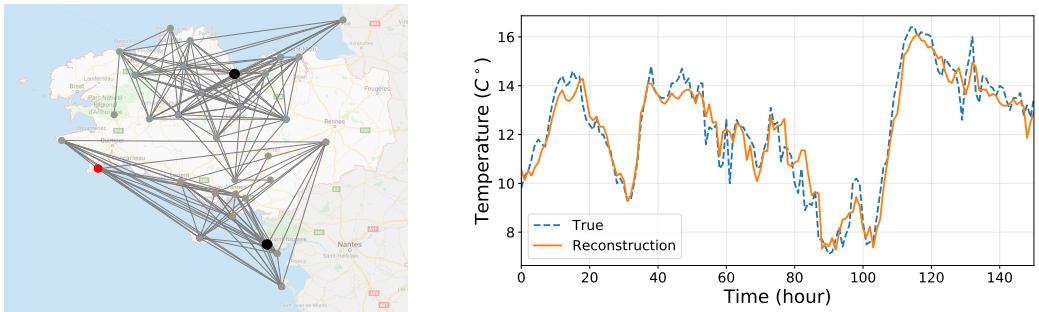
(c) Graph clustering on IGL-3SR's result.

(d) Graph clustering on FGL-3SR's result.

Figure 8: (Top row) Learned graph with (a) IGL-3SR and (b) FGL-3SR. The node color corresponds to the average temperature in C° during all the period of observation. (Bottom row) Graph segmentation in two parts (red vs. green nodes) with spectral clustering using the Laplacian matrix learned by (c) IGL-3SR and (d) FGL-3SR.

The graphs obtained with each of the method are displayed in Figure 8 (a-b). They are in accordance with the one found in Chepuri et al. (2017) on the same dataset. Both the proposed methods provide similar results, which shows that the relaxation used in FGL-3SR has a moderate influence in practice in this real-world problem. Although ground-truth is not available for this use-case, the quality of the learned graph can be assessed when using it as input in standard tasks such as graph clustering or sampling. For instance, when applying spectral clustering (Ng et al., 2001) with two clusters on the resulting Laplacian matrices, it can be seen that both methods split the learned graph in two parts corresponding to the north and the south of the region of Britanny (Figure 8 (c-d)), which is an expected natural segmentation.

The learned graphs can be also employed in the graph sampling task. Indeed, due to the constraints used in the optimization problem, the graph signals are bandlimited with respect to learned graphs. For instance, in this example the graph signals are 2-bandlimited. This property means that it is possible to select only 2 nodes and to reconstruct the graph signal values of the 30 remaining nodes using linear interpolation. Figure 9 displays an example of such reconstruction: thanks to the learned graph structure, the use of only 2



(a) **(b)**

Figure 9: (a) Two nodes (black) used for signal interpolation to infer the signal at a target node (red). (b) The true signal at the target node and its reconstruction from only the two selected nodes.

nodes allows to reconstruct sufficiently well the whole data matrix with a mean absolute error of 0.614. Again, this is a very interesting result that indirectly shows the quality of the learned graph.

7.5 Cancer genome data

In this second experiment, we consider the RNA-Seq Cancer Genome Atlas Research Network data (Weinstein et al., 2013). The dataset contains the information of 801 individuals, each of them is characterized by $20,531$ genetic features and labeled by one out of 5 types of cancer: breast carcinoma (BRCA), colon adenocarcinoma (COAD), kidney renal clear-cell carcinoma (KIRC), lung adenocarcinoma (LUAD), and prostate adenocarcinoma (PRAD).

The goal of the task is to learn a graph of the $N = 801$ individuals (nodes) using the $n = 20531$ genetic features (samples seen here as graph signals), and determine if this graph is able to group the individuals according to their tumor type. Since the number of nodes is large, we propose to use FGL-3SR and, as previously, spectral clustering (Ng et al., 2001) on the learned graph to find the cluster mapping.

As the number of nodes of the graph is too large, ESA-GL and GL-SigRep are not able to run in reasonable time. Therefore, we compare FGL-3SR to two other state-of-the-art methods, which are however not GSP-oriented but rather specialized to obtain a graph that facilitates data clustering. The two competitors are namely the Constrained Laplacian Rank (CLR) algorithm (Nie et al., 2016) that builds a special graph from the available data, and the Structured Graph Learning (SGL) algorithm (Kumar et al., 2019) that take as input the sample covariance matrix of the data. As quality measure we use the clustering accuracy, which has also been used in the associated papers of the competitors from where we obtain their reported results. The results for the three methods are respectively:

$$\text{FGL-3SR: } 0.9887, \qquad \text{CLR: } 0.9862, \qquad \text{SGL: } 0.9987.$$

The first interesting result is that FGL-3SR presents similar accuracy to CLR and SGL, even though it is not a graph learning method specially designed for clustering like the competitors. Secondly, while FGL-3SR comes second in terms of accuracy after SGL, two important remarks need to be made about the SGL method: 1) it must fix the right number of clusters of the learned graph a priori to obtain such result; 2) it has an additional

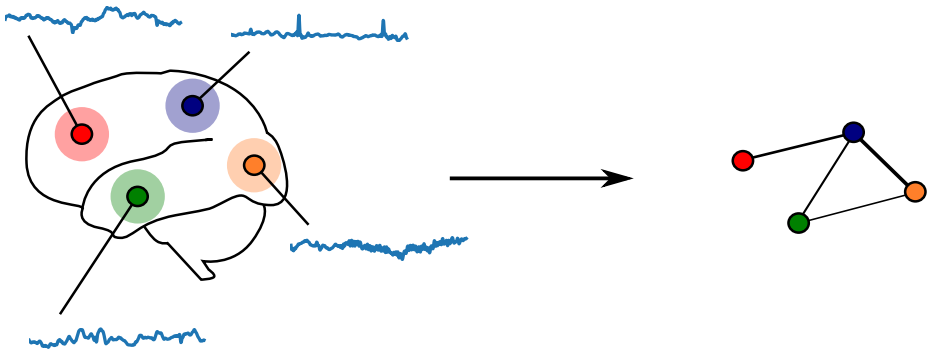


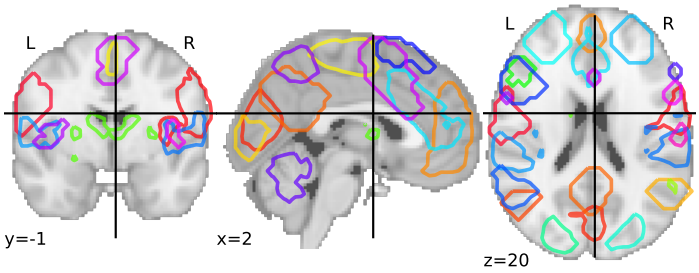
Figure 10: Schematic inference of functional connectivity between different regions of the brain. Left: time series recorded at different regions of the brain. Each signal corresponds to a local aggregation of the fMRI recorded in the corresponding ROI (node). Right: a functional connectivity graph where the nodes represent the brain regions and the edges represent the strength of functional connections between these regions.

hyperparameter to tune compared to FGL-3SR. Therefore comparably, bearing in mind the above results, the fact that SGL is fine-tailored for the undertaken clustering tasks and that it has higher tuning complexity, and finally the limitations of ESA-GL and Sig-Rep that prevent them from being applied in this scenario, FGL-3SR seems to be a promising alternative for large-scale graph-based learning applications.

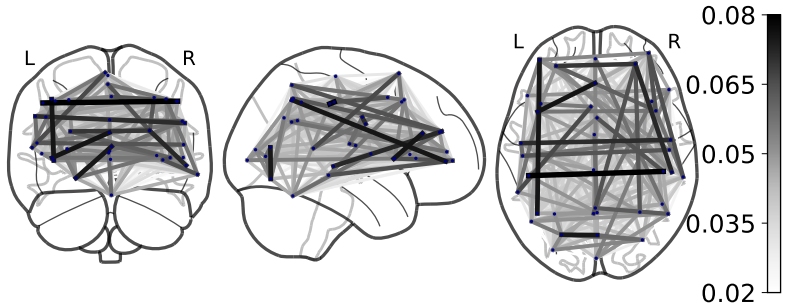
7.6 Results on the ADHD dataset

In this third experiment, we consider the Attention Deficit Hyperactivity Disorder (ADHD) dataset (Bellec et al., 2017) composed of functional Magnetic Resonance Imaging (fMRI) data. ADHD is a mental pathophysiology characterized by an excessive activity (Boyle et al., 2011). We study the resting-state fMRI of 20 subjects with ADHD and 20 healthy subjects available from `Nilearn` (Abraham et al., 2014). Each of the 40 fMRIs consists in a series of images measuring the brain activity. These images are processed as follows. The brain is first segmented into $N = 39$ Regions Of Interest (ROIs) via the Multi-Subject Dictionary Learning atlas (Varoquaux et al., 2011) (see Figure 11a). Each ROI is considered to define a graph node whose signal value is the aggregation of the ROI's pixel values. This way, each fMRI image is transformed into a graph signal. For each of the 40 subjects, we therefore have access to a matrix in $\mathbb{R}^{n\times 39}$, where n is the number of images in the fMRI of the subject (i.e. the number of graph signals).

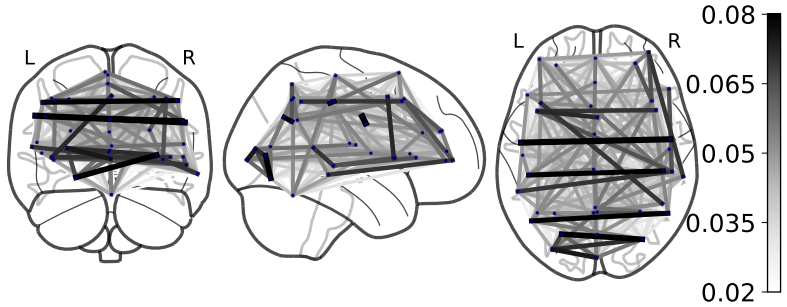
The considered task is to estimate a graph independently for each subject. A schematic illustration of the learning process is displayed in Figure 10. Examples of learned graphs with FGL-3SR for an ADHD subject, and a healthy subject, are displayed in Figure 11. Visually, they reveal strong symmetric links between the right and left hemisphere of the brain. This phenomenon is common in resting-state fMRI where one hemisphere tends to correlate highly with the homologous anatomical location in the opposite hemisphere (Damoiseaux et al., 2006; Smith et al., 2009). Pointing out differences, though, the graph from the ADHD subject seems less structured and contains several spurious links (diagonal and north-south connections).



(a) Indicative Regions of Interest (ROIs) from Varoquaux et al. (2011).



(b) ADHD subject.



(c) Healthy subject.

Figure 11: (a) Indicative ROIs from the Multi-Subject Dictionary Learning atlas extracted in Varoquaux et al. (2011) with sparse dictionary learning. Results: Graphs returned by FGL-3SR, separately for (b) an ADHD patient and (c) a healthy subject, where darker edges indicate larger weights of connection.

Aiming to better highlight the potential value of quality learned graphs for such studies, we proceed and use the Laplacian matrices of the brain graphs to classify the subjects, as proposed in several resting-state fMRI studies (Abraham et al., 2017; Dadi et al., 2019). First, we subtract the average graph for all subjects, which in fact removes the symmetrical connections common to all subjects), and then we use a 3-Nearest Neighbors classification algorithm. We use the correlation coefficient of Equation (24) as distance metric between Laplacian matrices, and a leave-one-out cross-validation strategy. The classification accuracy of the described approach reaches 65%. This level shall be compared with the performance obtained using simple correlation graphs (Abraham et al., 2017) that, on these 40 subjects, leads to an accuracy of 52.5%. It appears that in this context the use of a more sophisticated graph learning process allows a subject characterization that goes beyond

considering basic statistical correlation effects. Interestingly, this score is also comparable with state-of-the-art results reported in Sen et al. (2018) for the same task, but on a larger database (67.3% of accuracy), using more sophisticated and specially-tailored processing steps, as well as carefully chosen classifiers.

At this point, we should make absolutely clear that with this case study we only illustrate the general usefulness and potential of the graphs learned using our approach. Besides the lack of deeper clinical evaluation of the findings, there are two more technical reasons that would limit the value of any further comparison or conclusion. First, we do not know whether our assumptions (sparsity and bandlimitedness) are the best or even perfectly valid for the specific problem. Second, the small size of the data set limits the statistical importance of any comparative results on the basis of common evaluation metrics, such as the accuracy. Nevertheless, we believe that the reported result motivates future work on ADHD or other related disorders on the direction of exploring further the potential of sophisticated graph learning techniques.

8. Conclusions

This article presented a data-driven graph learning approach by employing a combination of two assumptions. The first is standard in the related literature and concerns the *smoothness* of graph signals with respect to the underlying graph structure. The second is the *spectral sparsity* assumption, a consequence of the presence of clusters in real-world graphs. We proposed two algorithms to solve the corresponding optimization problem. The first one, IGL-3SR, has the advantage to decrease the objective function at each iteration. To address its low speed of convergence, we propose FGL-3SR that is a fast and scalable alternative. The findings of our empirical evaluation on synthetic data showed that the proposed approaches are as good or better performing than the reference state-of-the-art algorithms in term of reconstructing the unknown underlying graph and of computational cost (running time). Experiments on real-world benchmark use-cases suggest that our algorithms learn graphs that are useful and promising for any graph-based machine learning methodology, such as graph clustering and subsampling, etc. Finally, by including the two assumptions in a probabilistic model, we link our optimization problem to a maximum a posteriori estimation and pave the way for further statistical understanding.

9. Technical proofs

This section provides the technical proofs of the different propositions exposed in the paper. Recall that lower case variables refer vectors/scalars while upper case variable denote matrices. The table below provides the main notations used in the technical discussion that that follows.

Lemma 5 – *Given $X, X_0 \in \mathbb{R}^{N\times N}$ two orthogonal matrices with first column equals to $\frac{1}{\sqrt{N}}\mathbf{1}_N$ (constraint (3a)), we have the following equality:*

$$X = X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\top \\ \mathbf{0}_{N-1} & [X_0^\top X]_{2:,2:} \end{bmatrix},$$

x^T, M^T	Transpose of vector x, matrix M.
$\mathrm{tr}(M)$	Trace of matrix M.
$\mathrm{diag}(x)$	Diagonal matrix containing the vector x.
$M_{k,l}$	(k,l)-th element of the matrix M.
$M_{k,:}$	k-th row of M.
$M_{:,l}$	l-th column of M.
$M_{k:,l:}$	Submatrix containing the elements of M from the k-th row to the last row, and from the l-th column to the last column.
$M \succeq 0$	M is a positive semi-definite matrix.
$M^\dagger$	The Moore-Penrose pseudoinverse of M.
e_k	Vector containing zeros except a 1 at position k.
I_n	Identity matrix of size n.
$\mathbf{0}_n$	Vector of size n containing only zeros.
$\mathbf{1}_n$	Vector of size n containing only ones.
$\mathbb{1}_A(\cdot)$	The indicator function over the set A.
$\|x\|_0$	The number of non-zero elements of a vector x.
$\|\cdot\|_F$	The Frobenius norm.
$\|\cdot\|_{2,0}$	The $\ell_{2,0}$-norm, with $\|M\|_{2,0} = \sum_{i=1} \mathbb{1}_{\{\|H_{i,:}\|_2 \neq 0\}}$.
$\|\cdot\|_{2,1}$	The $\ell_{2,1}$-norm, with $\|M\|_{2,1} = \sum_{i=1} \|H_{i,:}\|_2$.
∇f	Gradient of the function f.
$\langle \cdot, \cdot \rangle$	Inner product function.
$Orth(N)$	The set of all orthogonal matrices of size $N \times N$.
$\mathcal{O}(\,\cdot\,)$	Order of magnitude (e.g. of computational complexity).
τ	Number of iterations needed for an optimization procedure.

Table 3: Table of notations used throughout the article.

with $[X_0^\mathsf{T} X]_{2:,2:}$ denoting the submatrix of $X_0^\mathsf{T} X$ containing everything but the first row and column of itself. Furthermore, remark that $[X_0^\mathsf{T} X]_{2:,2:}$ is in $Orth(N-1)$.

Proof Let consider $X, X_0 \in \mathbb{R}^{N \times N}$ two orthogonal matrix with first column equals to $\frac{1}{\sqrt{N}}\mathbf{1}_N$. We have the following equalities:

$$X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & [X_0^\mathsf{T} X]_{2:,2:} \end{bmatrix} = \begin{bmatrix} \vdots & \vdots \\ X_{0(:,1)} & X_{0(:,2:)}[X_0^\mathsf{T} X]_{2:,2:} \\ \vdots & \vdots \end{bmatrix} = \begin{bmatrix} \vdots & \vdots \\ \frac{1}{\sqrt{N}}\mathbf{1}_N & X_{:,2:} \\ \vdots & \vdots \end{bmatrix} = X \ .$$

Furthermore, thanks to the orthogonality of X and X_0, we have

$$[X_0^\mathsf{T} X]_{2:,2:} \left[[X_0^\mathsf{T} X]_{2:,2:}\right]^\mathsf{T} = X_{0,(2:,:)}^\mathsf{T} X_{:,2:} \left[X_{0,(2:,:)}^\mathsf{T} X_{:,2:}\right]^\mathsf{T} = X_{0,(2:,:)}^\mathsf{T} X_{:,2:} X_{:,2:}^\mathsf{T} [X_{0,(2:,:)}^\mathsf{T}]^\mathsf{T} = I_{N-1}.$$

By symmetry we conclude that $[X_0^\mathsf{T} X]_{2:,2:} \in Orth(N-1)$. ■

Proposition 6 – *Given $X_0 \in \mathbb{R}^{N\times N}$ an orthogonal matrix with first column equals to $\frac{1}{\sqrt{N}}\mathbf{1}_N$, an equivalent formulation of optimization Problem* (3) *is given by:*

$$\min_{H,U,\Lambda} \left\| Y - X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U \end{bmatrix} H \right\|_F^2 + \alpha \|\Lambda^{1/2} H\|_F^2 + \beta \|H\|_S \triangleq f(H,U,\Lambda) \ ,$$

$$s.t. \quad \begin{cases} U^\mathsf{T} U = I_{N-1} \ , & \text{(a')} \\ \left(X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U \end{bmatrix} \Lambda \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U^\mathsf{T} \end{bmatrix} X_0^\mathsf{T} \right)_{k,\ell} \leq 0 \quad k \neq \ell \ , & \text{(b')} \\ \Lambda = diag(0, \lambda_2, ..., \lambda_N) \succeq 0 \ , & \text{(c)} \\ tr(\Lambda) = N \in \mathbb{R}_*^+ \ . & \text{(d)} \end{cases}$$

Proof From the previous lemma, we know that X can be decompose into two orthogonal matrices X_0 and $U = [X_0^\mathsf{T} X]_{2:,2:}$. Hence, we can optimize with respect to U instead of X and the second part of the constraint (3a) is automatically satisfied. To make the equivalence, we just replace X from the main optimization problem to $X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & U \end{bmatrix}$ where U is now imposed to be orthogonal. ■

Proposition 8 (Closed-form solution for the $\ell_{2,0}$ and $\ell_{2,1}$-norms) – *The solutions of Problem* (7) *when $\|\cdot\|_S$ is set to $\|\cdot\|_{2,0}$ or $\|\cdot\|_{2,1}$ are given in the following.*

- *Using the $\ell_{2,0}$-norm, the optimal solution of* (7) *is given by the matrix $\widehat{H} \in \mathbb{R}^{N\times n}$ where for $1 \leq i \leq N$,*

$$\widehat{H}_{i,:} = \begin{cases} 0 & \text{if } \|(X^\mathsf{T} Y)_{i,:}\|_2^2/(1+\alpha\lambda_i) \leq \beta \ , \\ (X^\mathsf{T} Y)_{i,:}/(1+\alpha\lambda_i) & \text{else} \ . \end{cases}$$

- *Using the $\ell_{2,1}$-norm, the optimal solution of* (7) *is given by the matrix $\widehat{H} \in \mathbb{R}^{N\times n}$ where for $1 \leq i \leq N$,*

$$\widehat{H}_{i,:} = \frac{1}{1+\alpha\lambda_i}\Big(1 - \frac{\beta}{2}\frac{1}{\|(X^\mathsf{T} Y)_{i,:}\|_2}\Big)_+ (X^\mathsf{T} Y)_{i,:} \ ,$$

where $(t)_+ \triangleq \max\{0,t\}$ is the positive part function.

Proof In the following, we suppose that $Y \neq 0$ since in this trivial case, the solution is simply given by $\widehat{H} = 0$.

Closed-form solution for the $\ell_{2,0}$. Recall that $\|H\|_{2,0} = \sum_{i=1} \mathbb{1}_{\{\|H_{i,:}\|_2 \neq 0\}}$, the objective function can be written as:

$$\begin{aligned}
f(X, \Lambda, H) &= \|X^\mathsf{T} Y - H\|_F^2 + \alpha \|\Lambda^{1/2} H\|_F^2 + \beta \|H\|_{2,0} \\
&= \|Y\|_F^2 + \sum_{i=1}^{N} \left(\sum_{j=1}^{n} \left(H_{i,j}^2 - 2(X^\mathsf{T} Y)_{i,j} H_{i,j} + \alpha \lambda_i H_{i,j}^2 \right) + \beta \mathbb{1}_{\{\|H_{i,:}\|_2 \neq 0\}} \right) \\
&= \|Y\|_F^2 + \sum_{i=1}^{N} \left(\|H_{i,:}\|_2^2 - 2\langle (X^\mathsf{T} Y)_{i,:}, H_{i,:} \rangle + \alpha \lambda_i \|H_{i,:}\|_2^2 + \beta \mathbb{1}_{\{\|H_{i,:}\|_2 \neq 0\}} \right) \\
&= \|Y\|_F^2 + \sum_{i=1}^{N} \left((1 + \alpha \lambda_i) \|H_{i,:}\|_2^2 - 2\langle (X^\mathsf{T} Y)_{i,:}, H_{i,:} \rangle + \beta \mathbb{1}_{\{\|H_{i,:}\|_2 \neq 0\}} \right) \\
&= \|Y\|_F^2 + \sum_{i=1}^{N} \tilde{f}_i(X, \Lambda, H_{i,:}) \; .
\end{aligned}$$

Our objective function is written as a sum of independent objective functions, each associated with a different $H_{i,:}$. Hence, we can optimize the problem for each i. Our problem for a given i is:

$$\min_{H_{i,:} \in \mathbb{R}^n} \quad (1 + \alpha \lambda_i) \|H_{i,:}\|_2^2 - 2\langle (X^\mathsf{T} Y)_{i,:}, H_{i,:} \rangle + \beta \mathbb{1}_{\{\|H_{i,:}\|_2 \neq 0\}} \; .$$

When we restrict the minimization to $\|H_{i,:}\|_2 = 0$, the unique solution is $\widehat{H}_{i,:} = \mathbf{0}_n$ and $\tilde{f}_i(X, \Lambda, \widehat{H}_{i,:}) = 0$.

When $\|H_{i,:}\|_2 \neq 0$, the objective function is convex and differentiable, thus it suffice to take the following derivative equal to 0:

$$\begin{aligned}
\frac{\partial}{\partial H_{i,:}} \tilde{f}_i(H_{i,:}) = 2(1 + \alpha \lambda_i) H_{i,:} - 2(X^\mathsf{T} Y)_{i,:} = 0 \; , \\
\widehat{H}_{i,:} = (X^\mathsf{T} Y)_{i,:} / (1 + \alpha \lambda_i) \; .
\end{aligned}$$

With this solution, the objective function $\tilde{f}_i$ is equal to:

$$\begin{aligned}
\tilde{f}(X, \Lambda, \widehat{H}_{i,:}) &= (1 + \alpha \lambda_i) \|(X^\mathsf{T} Y)_{i,:} / (1 + \alpha \lambda_i)\|_2^2 - 2\langle (X^\mathsf{T} Y)_{i,:}, (X^\mathsf{T} Y)_{i,:} / (1 + \alpha \lambda_i) \rangle + \beta \\
&= \frac{1}{1 + \alpha \lambda_i} \|(X^\mathsf{T} Y)_{i,:}\|_2^2 - \frac{2}{1 + \alpha \lambda_i} \|(X^\mathsf{T} Y)_{i,:}\|_2^2 + \beta \\
&= \beta - \frac{1}{1 + \alpha \lambda_i} \|(X^\mathsf{T} Y)_{i,:}\|_2^2 \; .
\end{aligned}$$

Hence, whenever $\frac{1}{1 + \alpha \lambda_i} \|(X^\mathsf{T} Y)_{i,:}\|_2^2 \leq \beta$, the objective function is positive, making $\widehat{H}_{i,:} = 0$ a better choice for the minimization and conversely. In conclusion, for all $1 \leq i \leq N$, the solution is:

$$\widehat{H}_{i,:} = \begin{cases} 0 & \text{if } \|(X^\mathsf{T} Y)_{i,:}\|_2^2 / (1 + \alpha \lambda_i) \leq \beta \; , \\ (X^\mathsf{T} Y)_{i,:} / (1 + \alpha \lambda_i) & \text{else .} \end{cases}$$

Closed-form solution for the $\ell_{2,1}$. Similarly to the $\ell_{2,0}$ case, the objective function can be decomposed by a sum of independent objectives functions.

$$\begin{aligned}
f(X,\Lambda,H) &= \|X^\mathsf{T}Y - H\|_F^2 + \alpha\|\Lambda^{1/2}H\|_F^2 + \beta\|H\|_{2,1} \\
&= \|Y\|_F^2 + \sum_{i=1}^{N}\left(\sum_{j=1}^{n}\Big(H_{i,j}^2 - 2(X^\mathsf{T}Y)_{i,j}H_{i,j} + \alpha\lambda_i H_{i,j}^2\Big) + \beta\sqrt{\sum_{j=1} H_{i,j}^2}\right) \\
&= \|Y\|_F^2 + \sum_{i=1}^{N}\left(\|H_{i,:}\|_2^2 - 2\langle(X^\mathsf{T}Y)_{i,:}, H_{i,:}\rangle + \alpha\lambda_i\|H_{i,:}\|_2^2 + \beta\|H_{i,:}\|_2\right) \\
&= \|Y\|_F^2 + \sum_{i=1}^{N}\left((1+\alpha\lambda_i)\|H_{i,:}\|_2^2 - 2\langle(X^\mathsf{T}Y)_{i,:}, H_{i,:}\rangle + \beta\|H_{i,:}\|_2\right) \\
&= \|Y\|_F^2 + \sum_{i=1}^{N}\tilde{f}_i(X,\Lambda,H_{i,:})\ .
\end{aligned}$$

Again, we can optimize the problem for each row i of H independently. Our problem for a given i is:

$$\min_{H_{i,:}\in\mathbb{R}^n} \quad (1+\alpha\lambda_i)\|H_{i,:}\|_2^2 - 2\langle(X^\mathsf{T}Y)_{i,:}, H_{i,:}\rangle + \beta\|H_{i,:}\|_2\ . \tag{25}$$

Although non-differentiable at $H_{i,:} = \mathbf{0}_n$, this function is convex and we need to find $H_{i,:}$ such that the vector $\mathbf{0}_n$ belongs to the subdifferential of $\tilde{f}_i$ denoted by $\partial\tilde{f}_i(H_{i,:})$ and is equal to:

$$\partial\tilde{f}_i(H_{i,:}) = \begin{cases} \mathcal{B}_2\big(-2(X^TY)_{i,:}, \beta\big) & \text{if } H_{i,:} = \mathbf{0}_n\ , \\ 2\big(1+\alpha\lambda_i + \dfrac{\beta}{2}\dfrac{1}{\|H_{i,:}\|_2}\big)H_{i,:} - 2(X^\mathsf{T}Y)_{i,:} & \text{otherwise}\ . \end{cases}$$

Where $\mathcal{B}_2$ stand for the ℓ_2-norm bowl.

Remark that when $\|(X^\mathsf{T}Y)_{i,:}\|_2 \leq \frac{\beta}{2}$, $\mathbf{0}_n \in \mathcal{B}_2\big(-2(X^TY)_{i,:}, \beta\big)$ and thus in this case $\widehat{H}_{i,:} = \mathbf{0}_n$.

On the contrary, when $\|(X^\mathsf{T}Y)_{i,:}\|_2 > \frac{\beta}{2}$, we must find $H_{i,:}$ such that:

$$\big(1+\alpha\lambda_i + \frac{\beta}{2}\frac{1}{\|H_{i,:}\|_2}\big)H_{i,:} = (X^\mathsf{T}Y)_{i,:}\ .$$

By tacking the norm of the previous equation, we obtain

$$\begin{aligned}
&\big(1+\alpha\lambda_i + \frac{\beta}{2}\frac{1}{\|H_{i,:}\|_2}\big)\|H_{i,:}\|_2 = \|(X^\mathsf{T}Y)_{i,:}\|_2 \\
\iff &(1+\alpha\lambda_i)\|H_{i,:}\|_2 + \frac{\beta}{2} = \|(X^\mathsf{T}Y)_{i,:}\|_2 \\
\iff &\|H_{i,:}\|_2 = \big(\|(X^\mathsf{T}Y)_{i,:}\|_2 - \frac{\beta}{2}\big)/(1+\alpha\lambda_i) > 0\ .
\end{aligned}$$

We can now replace $\|H_{i,:}\|_2$ in the initial equation and get $H_{i,:}$.

$$\big(1+\alpha\lambda_i+\frac{\beta(1+\alpha\lambda_i)}{2\|(X^\mathsf{T}Y)_{i,:}\|_2-\beta}\big)H_{i,:}=\frac{(1+\alpha\lambda_i)\|(X^\mathsf{T}Y)_{i,:}\|_2}{\|(X^\mathsf{T}Y)_{i,:}\|_2-\beta/2}H_{i,:}=(X^\mathsf{T}Y)_{i,:}$$
$$\iff H_{i,:}=\frac{\|(X^\mathsf{T}Y)_{i,:}\|_2-\beta/2}{(1+\alpha\lambda_i)\|(X^\mathsf{T}Y)_{i,:}\|_2}(X^\mathsf{T}Y)_{i,:}=\frac{1}{1+\alpha\lambda_i}\Big(1-\frac{\beta}{2}\frac{1}{\|(X^\mathsf{T}Y)_{i,:}\|_2}\Big)(X^\mathsf{T}Y)_{i,:}\,,$$

which concludes the proof. ■

Proposition 10 (Euclidean gradient with respect to U) – *The Euclidean gradient of f and ϕ with respect to U are*

$$\nabla_U f(H,U,\Lambda)=-2\big[(HY^\mathsf{T}X_0)_{2:,2:}\big]^\mathsf{T}+2U(HH^\mathsf{T})_{2:,2:}\,,$$
$$\nabla_U\phi(U,\Lambda)=-\sum_{k=1}^{N-1}\sum_{\ell>k}^{N}\frac{\Big(B_{k,\ell}+B_{k,\ell}^\mathsf{T}\Big)U\Lambda_{2:,2:}}{h(U,\Lambda)_{k,\ell}}\,.$$

with $\forall 1\le k,\ell\le N, B_{k,\ell}=\big(X_0^\mathsf{T}e_ke_\ell^\mathsf{T}X_0\big)_{2:,2:}$, and $h(\cdot)$ from Definition 7.

Proof We begin by computing the gradient of the main objective, with respect to U. Recall the objective function with respect to U:

$$f(H,U,\Lambda)=-2\mathrm{tr}(Y^\mathsf{T}X_0\begin{bmatrix}1 & \mathbf{0}_{N-1}^\mathsf{T}\\ \mathbf{0}_{N-1} & U\end{bmatrix}H)+\mathrm{tr}(H^\mathsf{T}\begin{bmatrix}1 & \mathbf{0}_{N-1}^\mathsf{T}\\ \mathbf{0}_{N-1} & U^\mathsf{T}U\end{bmatrix}H)\,.$$

The corresponding gradient is the following.

$$\begin{aligned}\nabla_U f(H,U,\Lambda)&=-2\nabla_U\mathrm{tr}(Y^\mathsf{T}X_0\begin{bmatrix}1 & \mathbf{0}_{N-1}^\mathsf{T}\\ \mathbf{0}_{N-1} & U\end{bmatrix}H)+\nabla_U\mathrm{tr}(H^\mathsf{T}\begin{bmatrix}1 & \mathbf{0}_{N-1}^\mathsf{T}\\ \mathbf{0}_{N-1} & U^\mathsf{T}U\end{bmatrix}H)\\
&=-2\nabla_U\mathrm{tr}\left(HY^\mathsf{T}X_0\begin{bmatrix}1 & \mathbf{0}_{N-1}^\mathsf{T}\\ \mathbf{0}_{N-1} & U\end{bmatrix}\right)+\nabla_U\mathrm{tr}\left(HH^\mathsf{T}\begin{bmatrix}1 & \mathbf{0}_{N-1}^\mathsf{T}\\ \mathbf{0}_{N-1} & U^\mathsf{T}U\end{bmatrix}\right)\\
&=-2\nabla_U\left((HY^\mathsf{T}X_0)_{1,1}\cdot 1+\mathrm{tr}\Big((HY^\mathsf{T}X_0)_{2:,2:}U\Big)\right)\\
&\quad+\nabla_U\left((HH^\mathsf{T})_{1,1}\cdot 1+\mathrm{tr}\big((HH^\mathsf{T})_{2:,2:}U^\mathsf{T}U\big)\right)\\
&=-2\big[(HY^\mathsf{T}X_0)_{2:,2:}\big]^\mathsf{T}+2U(HH^\mathsf{T})_{2:,2:}\,.\end{aligned}$$

We now derive the gradient of the barrier function $\phi(U,\Lambda)$ with respect to U:

$$\begin{aligned}\nabla_U\phi(U,\Lambda)&=-\sum_{k=1}^{N-1}\sum_{\ell>k}^{N}\nabla_U\log\Big(-h(U,\Lambda)_{k,\ell}\Big)\\
&=-\sum_{k=1}^{N-1}\sum_{\ell>k}^{N}\frac{1}{h(U,\Lambda)_{k,\ell}}\nabla_U h(U,\Lambda)_{k,\ell}\,.\end{aligned}$$

We can write the h function as:

$$\begin{aligned}
h(U,\Lambda)_{k,\ell} &= \left\langle e_k e_\ell^\intercal, h(U,\Lambda)\right\rangle = \left\langle X_0^\intercal e_k e_\ell^\intercal X_0, \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\intercal \\ \mathbf{0}_{N-1} & U \end{bmatrix} \Lambda \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\intercal \\ \mathbf{0}_{N-1} & U \end{bmatrix}^\intercal \right\rangle \\
&= \left\langle X_0^\intercal e_k e_\ell^\intercal X_0, \begin{bmatrix} \lambda_1 & \mathbf{0}_{N-1}^\intercal \\ \mathbf{0}_{N-1} & U\Lambda_{2:,2:}U^\intercal \end{bmatrix} \right\rangle = \operatorname{tr}\Big(X_0^\intercal e_\ell e_k^\intercal X_0 \begin{bmatrix} 0 & \mathbf{0}_{N-1}^\intercal \\ \mathbf{0}_{N-1} & U\Lambda_{2:,2:}U^\intercal \end{bmatrix} \Big) \\
&= \left(X_0^\intercal e_\ell e_k^\intercal X_0\right)_{1,1} \cdot 0 + \operatorname{tr}\Big(\left(X_0^\intercal e_\ell e_k^\intercal X_0\right)_{2:,2:} U\Lambda_{2:,2:}U^\intercal \Big) \\
&= \operatorname{tr}\Big(B_{k,\ell}^\intercal U\Lambda_{2:,2:}U^\intercal \Big) \, .
\end{aligned}$$

In conclusion we have $\nabla_U h(U,\Lambda)_{k,\ell} = \Big(B_{k,\ell} + B_{k,\ell}^\intercal\Big) U\Lambda_{2:,2:}$, which finishes the proof. ■

Before proving Lemma 11 and Theorem 12, let us prove the two following important Lemmas.

Lemma 19 – *Linear Independence Constraint Qualification (LICQ) holds for any $U \in Orth(N-1)$.*

Proof Let define the function relative to the orthogonality constraint $\varphi(\cdot)$:

$$\forall 1 \leq i \leq j \leq N, \quad \varphi_{i,j}(U) = \sum_{k=1}^{N-1} U_{k,i}U_{k,j} - I_{i,j} \, . \tag{26}$$

If $i = j$, $\varphi_{i,i}(U) = \sum_{k=1}^{N-1} U_{k,i}^2 - 1$. Otherwise, $\varphi_{i,j}(U) = \sum_{k=1}^{N-1} U_{k,i}U_{k,j}$. Therefore, we have that:

$$\nabla_U \varphi_{i,j}(U) = \begin{cases} [0_{N-1\times(i-1)}; 2U_{:,i}; 0_{(N-1)\times(N-1-i)}], & \text{if } i = j; \\ [0_{(N-1)\times(i-1)}; U_{:,j}; 0_{(N-1)\times(j-i-1)}; U_{:,i}; 0_{(N-1)\times(N-1-j)}], & \text{otherwise.} \end{cases} \tag{27}$$

We can see that $U_{:,i}$ from $\nabla\varphi_{i,i}(U)$ will only appear in the i-th column, but $U_{:,i}$ from $\nabla\varphi_{i,j}(U)$ with $i \neq j$ will not appear in the i-th column. Hence, each $\nabla\varphi_{i,j}(U)$ cannot be expressed as a linear combination of the others. Thus, the $\{\nabla\varphi_{i,j}(U)\}_{i,j}$ are linearly independent. ■

Lemma 11 (Optimality conditions) – *Karush-Khun-Tucker (KKT) conditions are necessary conditions for any optimal solution (H^*, U^*, Λ^*) of Problem 5.*

Proof The objective function and the constraint functions of Problem 5 are continuous and (sub-)differentiable. Constraints on Λ are affine i.e. the Linearity Constraint Qualification (LCQ) holds. Furthermore, from Lemma 19, the constraint on U verifies LICQ. This implies that any optimal solution (H^*, U^*, Λ^*) of Problem 5 has to satisfy the KKT conditions. ■

Lemma 20 – *KKT conditions are necessary conditions for any optimal solution $H^{'}$ (resp. $U^{'}$ and $\Lambda^{'}$) of the sub-problem in H (resp. U and Λ).*

Proof Same arguments used in the previous Lemma. ■

Theorem 12 (Convergence of IGL-3SR) – *The sequence $\{(H^{(\ell)}, U^{(\ell)}, \lambda^{(\ell)})\}_{\ell \geq 0}$ generated by Algorithm 1 admits at least one converging subsequence. If we also assume that each sub-problem is effectively minimized at each iteration, then, any accumulation point of the sequence satisfies the KKT conditions of Problem 5.*

Proof We begin the proof by showing that at least one sub-sequence of the sequence of iterates produces by the block coordinate descent of IGL-3SR $\{(H^{(\ell)}, U^{(\ell)}, \Lambda^{(\ell)})\}_{\ell \geq 0}$ is convergent.

We first observe that the objective function of Problem 5, denoted $\psi(\cdot)$, is lower-bounded over the set of constraints. Indeed, $f(H, U, \Lambda) \geq 0$ as a sum of norms and over the constraints set we also have

$$|h(U, \Lambda)_{kl}| \leq ||h(U, \Lambda)||_F \leq \mathrm{tr}(\Lambda) = N \ ,$$

which implies that $\phi(U, \lambda) \geq -\frac{1}{2} N(N-1)\log(N)$. The objective function value being non-increasing at each iteration ℓ, i.e. $\psi(H^{(\ell)}, U^{(\ell)}, \Lambda^{(\ell)}) \leq \psi(H^{(\ell-1)}, U^{(\ell-1)}, \Lambda^{(\ell-1)})$, the iterates of the objective function value, denoted $\{\psi^{(\ell)}\}_{\ell \geq 0}$ are ensured to converge to a limit point ψ^{∞}.

Let us now observe that given an initial point $(H^{(0)}, U^{(0)}, \Lambda^{(0)})$, the set of feasible points (H, U, Λ) satisfying $\psi(H, U, \Lambda) \leq \psi(H^{(0)}, U^{(0)}, \Lambda^{(0)})$ is compact. Furthermore, as $\forall \ell \geq 0$, $\psi^{(\ell)} \leq \psi^{(\ell-1)}$, the sequence $\{(H^{(\ell)}, U^{(\ell)}, \Lambda^{(\ell)})\}_{\ell \geq 0}$ belong to this set and by Bolzano-Weierstrass this implies that the sequence of iterates $\{(H^{(\ell)}, U^{(\ell)}, \Lambda^{(\ell)})\}_{\ell \geq 0}$ admits at least one converging subsequence. Note that since the objective function is continuous, the value of the objective at this limit point corresponds to ψ^{∞} (De Leeuw, 1994).

It remains now to show that any accumulation point $(H^{\infty}, U^{\infty}, \Lambda^{\infty})$ of the sequence $\{(H^{(\ell)}, U^{(\ell)}, \Lambda^{(\ell)})\}_{\ell \geq 0}$ satisfies the KKT conditions of Problem 5. The Lagrangian of Problem 5 is:

$$\mathcal{L}(H, U, \Lambda, M, a, \mu) = f(H, U, \Lambda) + \frac{1}{t}\phi(U, \Lambda) + \mathrm{tr}(M(U^{\mathsf{T}}U - I_{N-1})) + a(\sum_{i=2}^{N} \lambda_i - N) - \sum_{i=2}^{N} \mu_i \lambda_i \ ,$$

where $M \in \mathbb{R}^{(N-1)\times(N-1)}, a \in \mathbb{R}$ and $\mu \in \mathbb{R}^{N-1}$ are KKT multipliers. The KKT conditions are then given by:

$$\begin{aligned} & & \partial_{H,U,\Lambda}\mathcal{L}(H,U,\Lambda,M,a,\mu) \ni 0; \\ U^\mathsf{T}U = I_{N-1} \quad , \quad \sum_{i=2}^{N}\lambda_i = N \quad , \quad & \lambda_i \geq 0 \quad \forall i; \\ & \mu_i \geq 0 \quad \forall i; \\ & \sum_{i=2}^{N}\mu_i\lambda_i = 0; \end{aligned}$$

Let first introduce $z_1^{(\ell)} = (H^{(\ell+1)}, U^{(\ell)}, \Lambda^{(\ell)})$, $z_2^{(\ell)} = (H^{(\ell+1)}, U^{(\ell+1)}, \Lambda^{(\ell)})$, and $z_3^{(\ell)} = (H^{(\ell+1)}, U^{(\ell+1)}, \Lambda^{(\ell+1)})$. Let $\{(H^{(\ell_j)}, U^{(\ell_j)}, \Lambda^{(\ell_j)})\}_{j\geq 0}$ be a subsequence of $\{(H^{(\ell)}, U^{(\ell)}, \Lambda^{(\ell)})\}_{\ell\geq 0}$ converging to $(H^\infty, U^\infty, \Lambda^\infty)$. As the objective functions of the H-block and the Λ-block are uniquely minimize, from arguments of Tseng (2001) and Theorem 2 of Razaviyayn et al. (2013), we have $z_1^{(\ell_j)} \to (H^\infty, U^\infty, \Lambda^\infty)$, $z_2^{(\ell_j)} \to (H^\infty, U^\infty, \Lambda^\infty)$, and $z_3^{(\ell_j)} \to (H^\infty, U^\infty, \Lambda^\infty)$.

From Lemma 20, and because $z_1^{(\ell_j)}, z_2^{(\ell_j)}$, and $z_3^{(\ell_j)}$ converge to the same point $(H^\infty, U^\infty, \Lambda^\infty)$, we have that:
1) The matrix H^∞ satisfies the following KKT conditions of the H sub-problem (7):

$$\partial_H\mathcal{L}(H, U^\infty, \Lambda^\infty, M, a, \mu) = \partial_H f(H, U^\infty, \Lambda^\infty) \ni 0 \; .$$

2) There exist M^∞ such that the matrix U^∞ satisfies the following KKT conditions of the U sub-problem (11):

$$\begin{aligned} & \nabla_U\mathcal{L}(H^\infty, U, \Lambda^\infty, M^\infty, a, \mu) = \nabla_U f(H^\infty, U, \Lambda^\infty) + \frac{1}{t}\nabla_U\phi(U, \Lambda^\infty) + U(M^\infty + M^{\infty\mathsf{T}}) = 0 \\ & U^\mathsf{T}U = I_{N-1} \; . \end{aligned}$$

3) There exist Lagrange multipliers $\mu^\infty, a^\infty \in \mathbb{R}$ such that the matrix Λ^∞ satisfies the following KKT conditions of the Λ sub-problem (10):

$$\begin{aligned} & \nabla_\Lambda\mathcal{L}(H^\infty, U^\infty, \Lambda, a^\infty, \mu^\infty) = \nabla_\Lambda f(H^\infty, U^\infty, \Lambda) + \frac{1}{t}\nabla_\Lambda\phi(U^\infty, \Lambda) + \mathrm{diag}(a^\infty \mathbf{1}_{N-1} - \mu^\infty) = 0 \\ & \sum_{i=2}^{N}\lambda_i^\infty = N \quad , \quad \lambda_i^\infty \geq 0 \quad \forall i \\ & \mu_i^\infty \geq 0 \quad \forall i \\ & \sum_{i=2}^{N}\mu_i^\infty\lambda_i^\infty = 0 \; . \end{aligned}$$

Combining 1), 2) and 3) we can conclude that $(H^\infty, U^\infty, \Lambda^\infty)$ is a KKT solution of Problem 5 with Langrange multipliers M^∞, a^∞ and μ^∞. ■

Proposition 13 (Feasible eigenvalues) – *Given any $X \in \mathbb{R}^{N\times N}$ being an orthogonal matrix with first column equals to $1/\sqrt{N}$ (constraint (3a)), there always exist a matrix $\Lambda \in \mathbb{R}^{N\times N}$ such that the following constraints are satisfied:*

$$\begin{cases} (X\Lambda X^\mathsf{T})_{i,j} \leq 0 \quad i \neq j \ , & \text{(3b)} \\ \Lambda = diag(0, \lambda_2, ..., \lambda_N) \succeq 0 \ , & \text{(3c)} \\ tr(\Lambda) = c \in \mathbb{R}_*^+ \ . & \text{(3d)} \end{cases}$$

Proof Let us consider a positive real value $c > 0$. Taking $\Lambda = \text{diag}(0, c, ..., c)/(N-1)$ leads to $\text{tr}(\Lambda) = c$ and $\forall i \neq j, (X\Lambda X^\mathsf{T})_{i,j} = -c/N < 0$. However, this solution with constant eigenvalues actually corresponds to the complete graph. For our purpose, it is the worst case scenario as it contains no structural information between the nodes. ■

Proposition 14 (Closed-form solution of Problem (13)) – *Consider the optimization Problem* (13). *Let X_0 be any matrix that belongs to the constraints set (a), and $M = (X_0^\mathsf{T} Y H^\mathsf{T})_{2:,2:}$ the submatrix containing everything but the input's first row and first column. Finally, let PDQ^T be the SVD of M. Then, the problem admits the following closed form solution*

$$\widehat{X} = X_0 \begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & PQ^\mathsf{T} \end{bmatrix} \ .$$

Proof One can observe that the relaxed optimization problem is equivalent to finding:

$$\widehat{G} = \underset{G}{\text{argmin}} \left\| Y - X_0 \underbrace{\begin{bmatrix} 1 & \mathbf{0}_{N-1}^\mathsf{T} \\ \mathbf{0}_{N-1} & G \end{bmatrix}}_{\triangleq \tilde{G}} H \right\|_F^2 , \tag{28}$$

s.t. $G^\mathsf{T} G = I_{N-1}$. This is obtained by replacing X with $X_0\tilde{G}$.

Solving the above Equation (28) is equivalent to finding:

$$\widehat{G} = \arg\max_G \text{tr}\left(HY^\mathsf{T} X_0 \tilde{G}\right) = \arg\max_G \text{tr}\left(M^\mathsf{T} G\right),$$

s.t. $G^\mathsf{T}\widehat{G} = I_{N-1}$. Then, as proved in Zou et al. (2006), we finally have $G^* = PQ^\mathsf{T}$, which completes the proof. ■

Lemma 16 – *Assume the proposed Model (16). If $p_1 = 0$ and $p_i \in (0,1)$, $\forall i \geq 2$, then,*

$$\begin{aligned} -\log(p(h|y, X, \Lambda)) \propto \frac{1}{\sigma^2}\|y - Xh\|_2^2 + \frac{1}{2} h^\mathsf{T} \Lambda h \\ + \sum_{i=1}^{N} \mathbb{1}_{\{h_i \neq 0\}} \left(p_i \log(\frac{\lambda_i}{\sqrt{2\pi}}) - \log(p_i) - \log(\frac{\lambda_i}{\sqrt{2\pi}}) \right) \ . \end{aligned}$$

Proof Based on the Factor Analysis model and the independence of h_i's,

$$\begin{aligned}\log(p(h|y,X,\Lambda)) &\propto \log\left(p(y|h,X,\Lambda)\right) + \log\left(p(h|X,\Lambda)\right) \\ &\propto -\frac{1}{2\sigma^2}\|y-Xh\|_2^2 + \sum_{i=1}^{N}\log\left(p(h_i|\lambda_i)\right)\ . \end{aligned} \tag{29}$$

Let us now focus on $\log\left(p(h_i|\lambda_i)\right)$, for which we have:

$$\begin{aligned}\log\left(p(h_i|\lambda_i)\right) &= \log\left(\sum_{\gamma_i=\{0,1\}} p(h_i,\gamma_i|\lambda_i)\right) \\ &= \log\left(\sum_{\gamma_i=\{0,1\}} p(h_i,\gamma_i|\lambda_i)\frac{p(\gamma_i|h_i,\lambda_i)}{p(\gamma_i|h_i,\lambda_i)}\right) \\ &\underset{(=)}{\geq} \sum_{\gamma_i=\{0,1\}} p(\gamma_i|h_i,\lambda_i)\log\left(\frac{p(h_i,\gamma_i|\lambda_i)}{p(\gamma_i|h_i,\lambda_i)}\right)\ . \end{aligned}$$

The last equality is obtain using the concavity of the logarithm and Jensen inequality. For this particular case, it correspond to an equality. Then we have:

$$\begin{aligned}\log\left(p(h_i|\lambda_i)\right) = &\sum_{\gamma_i=\{0,1\}} p(\gamma_i|h_i,\lambda_i)\log\left(p(h_i,\gamma_i|\lambda_i)\right) \quad (\star) \\ &- \sum_{\gamma_i=\{0,1\}} p(\gamma_i|h_i,\lambda_i)\log\left(p(\gamma_i|h_i,\lambda_i)\right)\ . \quad (\star\star)\end{aligned}$$

Before computing the previous two sums, we need to observe that:

$$p(\gamma_i=1|h_i) = \begin{cases} 1 & \text{if} \quad h_i \neq 0\ , \\ p_i & \text{if} \quad h_i = 0\ . \end{cases}$$

We can now compute $(\star)$ and $(\star\star)$ as follows:

$$\begin{aligned}(\star) &= \sum_{\gamma_i=\{0,1\}} p(\gamma_i|h_i,\lambda_i)\left[\log\left(p(h_i|\gamma_i,\lambda_i)\right) + \log\left(p(\gamma_i|\lambda_i)\right)\right] \\ &= \left(\mathbb{1}_{\{h_i\neq 0\}} + p_i\mathbb{1}_{\{h_i=0\}}\right)\left[\log\left(\frac{\lambda_i}{\sqrt{2\pi}}\right) - \frac{1}{2}\lambda_i h_i^2 + \log\left(p_i\right)\right] \\ &\qquad + \left((1-p_i)\mathbb{1}_{\{h_i=0\}}\right)\left[\log\left(\mathbb{1}_{\{h_i=0\}}\right) + \log\left(1-p_i\right)\right] \\ (\star\star) &= \left[p_i\log(p_i) + (1-p_i)\log(1-p_i)\right]\mathbb{1}_{\{h_i=0\}}\ . \end{aligned}$$

Finally we can compute $\log\left(p(h_i|\lambda_i)\right)$:

$$\begin{aligned}
\log\left(p(h_i|\lambda_i)\right) &= (\star) - (\star\star) \\
&= \mathbb{1}_{\{h_i\neq 0\}}\left(\log\left(\frac{\lambda_i}{\sqrt{2\pi}}\right) - \frac{1}{2}\lambda_i h_i^2 + \log\left(p_i\right)\right) + p_i \log\left(\frac{\lambda_i}{\sqrt{2\pi}}\right)\mathbb{1}_{\{h_i=0\}} \\
&= \mathbb{1}_{\{h_i\neq 0\}}\left(\log\left(\frac{\lambda_i}{\sqrt{2\pi}}\right) + \log\left(p_i\right) - p_i\log\left(\frac{\lambda_i}{\sqrt{2\pi}}\right)\right) + p_i\log\left(\frac{\lambda_i}{\sqrt{2\pi}}\right) + -\frac{1}{2}\lambda_i h_i^2 \\
&\propto \mathbb{1}_{\{h_i\neq 0\}}\left(\log\left(\frac{\lambda_i}{\sqrt{2\pi}}\right) + \log\left(p_i\right) - p_i\log\left(\frac{\lambda_i}{\sqrt{2\pi}}\right)\right) + -\frac{1}{2}\lambda_i h_i^2 \ .
\end{aligned}$$

Note that with our parametrization, the particular case $i = 1$ leads to $\log\left(p(h_1|\lambda_1)\right) = 0$. Now plugging our result in Equation (29) and multiplying on both side by -1, we get our final result. ■

Proposition 18 (A posteriori distribution of h) – *Let $C > 0$, and assume for all $i \geq 2$ that $p_i = e^{-C}$ if $\lambda_i = \sqrt{2\pi}$ and $p_i = -W\left(-\frac{e^{-C}\log(\lambda_i/\sqrt{2\pi})}{\lambda_i/\sqrt{2\pi}}\right)/\log(\lambda_i/\sqrt{2\pi})$ if not. Then, $p_i \in (0,1)$ and there exist constants $\alpha, \beta > 0$ such that:*

$$-\log(p(h|y,X,\Lambda)) \propto \|y - Xh\|_2^2 + \alpha h^\mathsf{T}\Lambda h + \beta\|h\|_0 \ .$$

Proof To show that the p_i's are well-defined and belongs to $(0,1)$, it suffices to apply Lemma 21 with $x = \lambda_i/\sqrt{2\pi}$.

We now proof the main result of the proposition. If $\lambda_i = \sqrt{2\pi}$, then $p_i = e^{-C} < 1$ and

$$p_i\log(\frac{\lambda_i}{\sqrt{2\pi}}) - \log(p_i) - \log(\frac{\lambda_i}{\sqrt{2\pi}}) = -\log(p_i) = C \ .$$

If $\lambda_i \neq \sqrt{2\pi}$, then $-p_i\log(\lambda_i/\sqrt{2\pi}) = W\left(-\frac{e^{-C}\log(\lambda_i/\sqrt{2\pi})}{\lambda_i/\sqrt{2\pi}}\right)$. Since W corresponds to the inverse function of $f(W) = We^W$, we have:

$$\begin{aligned}
&-p_i\log(\lambda_i/\sqrt{2\pi})e^{-p_i\log(\lambda_i/\sqrt{2\pi})} = -\frac{e^{-C}\log(\lambda_i/\sqrt{2\pi})}{\lambda_i/\sqrt{2\pi}} \\
\iff &\left|-p_i\log(\lambda_i/\sqrt{2\pi})e^{-p_i\log(\lambda_i/\sqrt{2\pi})}\right| = \left|-\frac{e^{-C}\log(\lambda_i/\sqrt{2\pi})}{\lambda_i/\sqrt{2\pi}}\right| \\
\iff &\log\left(p_i\left|\log(\lambda_i/\sqrt{2\pi})\right|e^{-p_i\log(\lambda_i/\sqrt{2\pi})}\right) = \log\left(\frac{e^{-C}\left|\log(\lambda_i/\sqrt{2\pi})\right|}{\lambda_i/\sqrt{2\pi}}\right) \\
\iff &\log(p_i) + \log\left(\left|\log(\lambda_i/\sqrt{2\pi})\right|\right) - p_i\log(\lambda_i/\sqrt{2\pi}) \\
&\quad = -C + \log\left(\left|\log(\lambda_i/\sqrt{2\pi})\right|\right) - \log(\lambda_i/\sqrt{2\pi}) \ .
\end{aligned}$$

Same as the case where $\lambda_i = \sqrt{2\pi}$, the final equality gives us:

$$p_i \log(\frac{\lambda_i}{\sqrt{2\pi}}) - \log(p_i) - \log(\frac{\lambda_i}{\sqrt{2\pi}}) = C \; . \tag{30}$$

Plugging the Equation (30) into the final result of proposition 1, we obtain:

$$\begin{aligned} -\log(p(h|y, X, \Lambda)) &\propto \frac{1}{2\sigma^2}\|y - Xh\|_2^2 + \frac{1}{2}h^\mathsf{T}\Lambda h + C\|h\|_0 \\ &\propto \|y - Xh\|_2^2 + \alpha h^\mathsf{T}\Lambda h + \beta\|h\|_0 \; , \end{aligned}$$

taking $\alpha = \sigma^2$ and $\beta = 2C\sigma^2$. This concludes the proof. ■

Lemma 21 *Let $C > 0$. For any $x > 0$,*

$$0 \leq -W\left(-\frac{e^{-C}\log(x)}{x}\right) / \log(x) \leq 1 \; . \tag{31}$$

Proof First, we show that this function is decreasing for $x > 0$. The derivative of the function is given by

$$\frac{\partial}{\partial x} - W\left(-\frac{e^{-C}\log(x)}{x}\right) / \log(x) = \frac{W\left(-\frac{e^{-C}\log(x)}{x}\right)\left(W\left(-\frac{e^{-C}\log(x)}{x}\right) + \log(x)\right)}{x\log^2(x)\left(W\left(-\frac{e^{-C}\log(x)}{x}\right) + 1\right)} \; . \tag{32}$$

For $x > 0$ and $C > 0$,

$$-1/e < -e^{-(C+1)} = \min_{x>0} -\frac{e^{-C}\log(x)}{x} \leq -\frac{e^{-C}\log(x)}{x} \; . \tag{33}$$

As $W(\cdot)$ is strictly increasing for $x > -1/e$, we have $W\left(-\frac{e^{-C}\log(x)}{x}\right) > W\left(-1/e\right) = -1$. Hence, the bottom part of the previous equation is always positive.

For $0 < x \leq 1$, $W\left(-\frac{e^{-C}\log(x)}{x}\right)$ is positive. Furthermore,

$$-e^{-C}\frac{\log(x)}{x} < -\frac{\log(x)}{x} \iff W\left(-e^{-C}\frac{\log(x)}{x}\right) < W\left(\frac{-\log(x)}{x}\right) = -\log(x) \tag{34}$$

$$\iff W\left(-e^{-C}\frac{\log(x)}{x}\right) + \log(x) < 0 \; . \tag{35}$$

Hence, when $0 < x \leq 1$, the upper part of the previous equation is negative.

For $1 < x \leq e$, $W\left(-\frac{e^{-C}\log(x)}{x}\right)$ is negative. Furthermore,

$$-\frac{1}{e} \leq -\frac{\log(x)}{x} < -e^{-C}\frac{\log(x)}{x} \iff W\left(-\frac{\log(x)}{x}\right) = -\log(x) < W\left(-e^{-C}\frac{\log(x)}{x}\right) \tag{36}$$

$$\iff W\left(-e^{-C}\frac{\log(x)}{x}\right) + \log(x) > 0 \; . \tag{37}$$

Hence, when $1 < x \leq e$, the upper part of the previous equation is negative again.

For $x > e$, $W\left(-\frac{e^{-C}\log(x)}{x}\right)$ is negative. Furthermore, $W\left(-\frac{e^{-C}\log(x)}{x}\right) > -1$ and $\log(x) > 1$. Hence, the addition is positive and the upper part of the previous equation is negative again.

We just have shown that the derivative is negative for $x > 0$. Hence, the initial function is decreasing on this interval. We now go back to the initial inequality (31). The left part of the inequality is straightforward as for x large enough, the function corresponds to the product of two positive functions. The function being decreasing, the lower bound follows. For the upper bound, let us remind that for $y > e$, we have the inequality $W(y) < \log(y)$ (Hoorfar and Hassani, 2007). Let $f(x) = -\frac{e^{-C}\log(x)}{x}$, for x small enough we have:

$$\begin{aligned} W\left(f(x)\right) < \log(f(x)) \quad &\Leftrightarrow \quad -W\left(f(x)\right) > -\log(f(x)) \\ &\Leftrightarrow -W\left(f(x)\right)/\log(x) < -\log(f(x))/\log(x)\ . \end{aligned}$$

Tacking the limit when $x \to 0_+$ conclude the proof,

$$\begin{aligned} \lim_{x\to 0_+} -\log(f(x))/\log(x) &= \lim_{x\to 0_+} -\log(-\frac{e^{-C}\log(x)}{x})/\log(x) \\ &= \lim_{x\to 0_+} -\Big(\log(e^{-C}) + \log(-\log(x)) - \log(x)\Big)/\log(x) \\ &= \lim_{x\to 0_+} \frac{C}{\log(x)} + \frac{\log(\log(1/x))}{\log(1/x)} + 1 = 1\ . \end{aligned}$$

■

References

A. Abraham, F. Pedregosa, M. Eickenberg, P. Gervais, A. Mueller, J. Kossaifi, A. Gramfort, B. Thirion, and G. Varoquaux. Machine learning for neuroimaging with scikit-learn. *Frontiers in Neuroinformatics*, 8:14, 2014.

A. Abraham, M. P. Milham, A. Di Martino, R. C. Craddock, D. Samaras, B. Thirion, and G. Varoquaux. Deriving reproducible biomarkers from multi-site resting-state data: an autism-based example. *NeuroImage*, 147:736–745, 2017.

P.-A. Absil, R. Mahony, and R. Sepulchre. *Optimization algorithms on matrix manifolds*. Princeton University Press, 2009.

A. Anis, A. Gadde, and A. Ortega. Towards a sampling theorem for signals on arbitrary graphs. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3864–3868, 2014.

A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In *Advances in Neural Information Processing Systems*, pages 41–48, 2007.

R. Arora. On learning rotations. In *Advances in Neural Information Processing Systems*, pages 55–63, 2009.

O. Banerjee, L. E. Ghaoui, and A. d'Aspremont. Model selection through sparse maximum likelihood estimation for multivariate Gaussian or binary data. *Journal of Machine Learning Research*, 9:485–516, 2008.

P. Bellec, C. Chu, F. Chouinard-Decorte, Y. Benhajali, D. S. Margulies, and R. C. Craddock. The neuro bureau ADHD-200 preprocessed repository. *Neuroimage*, 144:275–286, 2017.

S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

S. Boyd and L. Vandenberghe. *Introduction to applied linear algebra: vectors, matrices, and least squares*. Cambridge University Press, 2018.

C. A. Boyle, S. Boulet, L. A. Schieve, R. A. Cohen, S. J. Blumberg, M. Yeargin-Allsopp, S. Visser, and M. D. Kogan. Trends in the prevalence of developmental disabilities in US children, 1997–2008. *Pediatrics*, 127(6):1034–1042, 2011.

S. Bubeck. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.

S. Chen, A. Sandryhaila, and J. Kovačević. Sampling theory for graph signals. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3392–3396, 2015.

S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero. Learning sparse graphs under smoothness prior. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6508–6512, 2017.

F. R. K. Chung and F. C. Graham. *Spectral graph theory*. Number 92. American Mathematical Society, 1997.

A. K. Cline and I. S. Dhillon. Computation of the singular value decomposition. 2006.

K. Dadi, M. Rahim, A. Abraham, D. Chyzhyk, M. Milham, B. Thirion, G. Varoquaux, Alzheimer's Disease Neuroimaging Initiative, et al. Benchmarking functional connectome-based predictive models for resting-state fMRI. *Neuroimage*, 192:115–134, 2019.

S. I. Daitch, J. A. Kelner, and D. A. Spielman. Fitting a graph to vector data. In *Proceedings of the International Conference on Machine Learning*, pages 201–208, 2009.

J. S. Damoiseaux, S. Rombouts, F. Barkhof, P. Scheltens, C. J. Stam, S. M. Smith, and C. F. Beckmann. Consistent resting-state networks across healthy subjects. *Proceedings of the national academy of sciences*, 103(37):13848–13853, 2006.

Jan De Leeuw. Block-relaxation algorithms in statistics. In *Information systems and data analysis*, pages 308–324. Springer, 1994.

S. Diamond and S. Boyd. Cvxpy: A python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17:1–5, 2016.

P. M. Djuric and C. Richard, editors. *Cooperative and Graph Signal Processing – Principles and Applications*. Elsevier, 2018.

X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Learning Laplacian matrix in smooth graph signal representations. *IEEE Transactions on Signal Processing*, 64(23): 6160–6173, 2016.

X. Dong, D. Thanou, M. Rabbat, and P. Frossard. Learning graphs from data: A signal representation perspective. *preprint arXiv:1806.00848*, 2018.

N. Du, L. Song, M. Yuan, and A. J. Smola. Learning networks of heterogeneous influence. In *Advances in Neural Information Processing Systems*, pages 2780–2788, 2012.

J. Duchi, S. Shalev-Shwartz, Y. Singer, and T. Chandra. Efficient projections onto the l 1-ball for learning in high dimensions. In *Proceedings of the International Conference on Machine Learning*, pages 272–279, 2008.

A. Edelman, T. A. Arias, and S. T. Smith. The geometry of algorithms with orthogonality constraints. *SIAM Journal on Matrix Analysis and Applications*, 20(2):303–353, 1998.

H. E. Egilmez, E. Pavez, and A. Ortega. Graph learning from data under structural and Laplacian constraints. *preprint arXiv:1611.05181*, 2016.

J. Friedman, T. Hastie, and R. Tibshirani. Sparse inverse covariance estimation with the graphical lasso. *Biostatistics*, 9(3):432–441, 2008.

M. Gomez-Rodriguez, L. Song, H. Daneshmand, and B. Schölkopf. Estimating diffusion networks: Recovery conditions, sample complexity & soft-thresholding algorithm. *Journal of Machine Learning Research*, 17:3092–3120, 2016.

Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

M. Hecker, S. Lambeck, S. Toepfer, E. Van Someren, and R. Guthke. Gene regulatory network inference: data integration in dynamic models — A review. *Biosystems*, 96(1): 86–103, 2009.

A. Hoorfar and M. Hassani. Approximation of the lambert w function and hyperpower function. *Research report collection*, 10(2), 2007.

A. Hyvärinen and E. Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4-5):411–430, 2000.

V. Kalofolias. How to learn a graph from smooth signals. In *Proceedings of the Conference on Artificial Intelligence and Statistics*, pages 920–929, 2016.

D. Koller, N. Friedman, and F. Bach. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Sandeep Kumar, Jiaxi Ying, Jos'e Vin'icius de Cardoso, Daniel P Palomar, et al. Structured graph learning via laplacian spectral constraints. In *Advances in Neural Information Processing Systems*, 2019.

B. Le Bars, P. Humbert, L. Oudre, and A. Kalogeratos. Learning Laplacian matrix from bandlimited graph signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2937–2941, 2019.

Batiste Le Bars, Pierre Humbert, Argyris Kalogeratos, and Nicolas Vayatis. Learning the piece-wise constant graph structure of a varying ising model. In *International Conference on Machine Learning*, pages 675–684. PMLR, 2020.

Kuzma Leshakov. F3: Fast forest fire graph generation. In *Companion to the first International Conference on the Art, Science and Engineering of Programming*, pages 1–3, 2017.

Jure Leskovec and Rok Sosič. Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1):1–20, 2016.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graphs over time: densification laws, shrinking diameters and possible explanations. In *Proceedings of the eleventh ACM international conference on Knowledge discovery in data mining (SIGKDD)*, pages 177–187, 2005.

Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM transactions on Knowledge Discovery from Data (TKDD)*, 1 (1):2–es, 2007.

Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: an approach to modeling networks. *Journal of Machine Learning Research*, 11(2), 2010.

P.-E. Maingé. Strong convergence of projected subgradient methods for nonsmooth and nonstrictly convex minimization. *Set-Valued Analysis*, 16(7-8):899–912, 2008.

A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro. Sampling of graph signals with successive local aggregations. *IEEE Transactions on Signal Processing*, 64(7):1832–1843, 2016.

G. Meyer. *Geometric optimization algorithms for linear regression on fixed-rank matrices*. PhD thesis, 2011.

S.K. Narang, A. Gadde, and A. Ortega. Signal processing techniques for interpolation in graph structured data. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5445–5449, 2013.

A.Y. Ng, M.I. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems*, pages 849–856, 2001.

Feiping Nie, Xiaoqian Wang, Michael I Jordan, and Heng Huang. The constrained laplacian rank algorithm for graph-based clustering. In *AAAI Conference on Artificial Intelligence*, 2016.

B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat. Characterization and inference of graph diffusion processes from observations of stationary signals. *IEEE Transactions on Signal and Information Processing over Networks*, 2017.

S. Ravishankar and Y. Bresler. Learning sparsifying transforms. *IEEE Transactions on Signal Processing*, 61(5):1072–1086, 2012.

Meisam Razaviyayn, Mingyi Hong, and Zhi-Quan Luo. A unified convergence analysis of block successive minimization methods for nonsmooth optimization. *SIAM Journal on Optimization*, 23(2):1126–1153, 2013.

M. G. Rodriguez, D. Balduzzi, and B. Schölkopf. Uncovering the temporal dynamics of diffusion networks. In *Proceedings of the International Conference on Machine Learning*, pages 561–568, 2011.

S. Sardellitti, S. Barbarossa, and P. Di Lorenzo. Graph topology inference based on sparsifying transform learning. *IEEE Transactions on Signal Processing*, 67(7):1712–1727, 2019.

B. Sen, N. C Borle, R. Greiner, and M. R. G. Brown. A general prediction model for the detection of ADHD and autism using structural and functional MRI. *PloS one*, 13(4): e0194856, 2018.

U. Shalit and G. Chechik. Coordinate-descent for learning orthogonal matrices through givens rotations. In *International Conference on Machine Learning*, pages 548–556, 2014.

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. *Signal Processing Magazine*, 30(3):83–98, 2013.

S. M. Smith, P. T. Fox, K. L. Miller, D. C. Glahn, P. M. Fox, C. E. Mackay, N. Filippini, K. E. Watkins, R. Toro, A. R. Laird, et al. Correspondence of the brain's functional architecture during activation and rest. *Proceedings of the National Academy of Sciences*, 106(31):13040–13045, 2009.

Christian L Staudt, Aleksejs Sazonovs, and Henning Meyerhenke. Networkit: A tool suite for large-scale complex network analysis. *Network Science*, 4(4):508–530, 2016.

D. A. Tarzanagh and G. Michailidis. Estimation of graphical models through structured norm minimization. *Journal of Machine Learning Research*, 18, 2018.

D. Thanou, X. Dong, D. Kressner, and P. Frossard. Learning heat diffusion graphs. *IEEE Transactions on Signal and Information Processing over Networks*, 3(3):484–499, 2017.

J. Townsend, N. Koep, and S. Weichwald. Pymanopt: A python toolbox for optimization on manifolds using automatic differentiation. *Journal of Machine Learning Research*, 17: 1–5, 2016.

Paul Tseng. Convergence of a block coordinate descent method for nondifferentiable minimization. *Journal of optimization theory and applications*, 109(3):475–494, 2001.

D. Valsesia, G. Fracastoro, and E. Magli. Sampling of graph signals via randomized local aggregations. *preprint arXiv:1804.06182*, 2018.

L. Vandenberghe. The CVXOPT linear and quadratic cone program solvers. 2010.

G. Varoquaux, A. Gramfort, F. Pedregosa, V. Michel, and B. Thirion. Multi-subject dictionary learning to segment an atlas of brain spontaneous activity. In *Proceedings of the Biennial International Conference on Information Processing in Medical Imaging*, pages 562–573. Springer, 2011.

U. Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

J. Wang and M. Kolar. Inference for high-dimensional exponential family graphical models. In *Artificial Intelligence and Statistics*, pages 1042–1050, 2016.

John N Weinstein, Eric A Collisson, Gordon B Mills, Kenna R Mills Shaw, Brad A Ozenberger, Kyle Ellrott, Ilya Shmulevich, Chris Sander, Joshua M Stuart, Cancer Genome Atlas Research Network, et al. The cancer genome atlas pan-cancer analysis project. *Nature genetics*, 45(10):1113, 2013.

L. H. William, Y. Rex, and J. Leskovec. Representation learning on graphs: Methods and applications. *preprint arXiv:1709.05584*, 2017.

E. Yang, P. Ravikumar, G. I. Allen, and Z. Liu. Graphical models via univariate exponential family distributions. *Journal of Machine Learning Research*, 16:3813–3847, 2015.

M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, 2006.

H. Zou, T. Hastie, and R. Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006.