Learning Graphs from Signal Observations under Smoothness Prior

Xiaowen Dong, Dorina Thanou, Pascal Frossard, and Pierre Vandergheynst

Abstract

The construction of a meaningful graph plays a crucial role in the success of many graph-based data representations and algorithms, especially in the emerging field of signal processing on graphs. However, a meaningful graph is not always readily available from the data, nor easy to define depending on the application domain. In this paper, we address the problem of graph learning, where we are interested in learning graph topologies, namely, the relationships between data entities, that well explain the signal observations. In particular, we want to infer a graph such that the input data forms graph signals with smooth variations on the resulting topology. To this end, we adopt a factor analysis model for the graph signals and impose a Gaussian probabilistic prior on the latent variables that control these graph signals. We show that the Gaussian prior leads to an efficient representation that favors the smoothness property of the graph signals. We then propose an algorithm for learning graphs that enforce such smoothness property for the signal observations by minimizing the variations of the signals on the learned graph. Experiments on both synthetic and real world data demonstrate that the proposed graph learning framework can efficiently infer meaningful graph topologies from only the signal observations.

Index Terms

Graph learning, signal processing on graphs, representation theory, factor analysis, Gaussian prior.

I. INTRODUCTION

Modern data processing tasks often manipulate structured data, where signal values are defined on the vertex set $V$ of a weighted and undirected graph $G$ with $n$ vertices. We refer to such data as graph
signals, where the vertices of the graph represent the entities and the edge weights reflect the pairwise relationships between these entities. The signal values associated with the vertices carry the information of interest in observations or physical measurements. Mathematically, a graph signal can be represented by a function $f : V \rightarrow \mathbb{R}^n$ which assigns a scalar value to each vertex (entity). Numerous examples can be found in real world applications, such as temperatures within a geographical area, transportation capacities at hubs in a transportation network, or human behaviors in a social network. Such data representations have led to the emerging field of signal processing on graphs [1], [2], where people are interested in the representation, processing and approximation of such structured signals. Currently, most of the research effort in this field has been devoted to the analysis and processing of the signals, which are defined on a graph that is a priori known or naturally chosen from the application domain, e.g., geographical or social friendship graphs. However, these choices of graphs that seem natural may not necessarily explain well the intrinsic relationships between the entities in the data and may therefore lead to suboptimal performance in graph signal processing and analysis tasks. Furthermore, it could be the case in some applications that a natural graph is not easy to define at all. In all these scenarios where a good graph is not readily available, we need to learn the graph topology from the observed data such that it captures the intrinsic relationships between the entities. This is exactly the motivation and main objective of the present paper.

The key challenge in the problem of graph learning is to compute the topology such that the relationships between the signals and the graph topology satisfy some pre-defined models. More precisely, we need some meaningful criteria to evaluate these relationships, which guide the graph learning process given the observed signals. In particular, we are interested in the family of signals that are smooth on graphs. This can be further motivated from the following simple example. Consider a signal given by a set of unordered scalar values, which can potentially live on three different graphs $G_1$, $G_2$ and $G_3$, leading to three graph signals shown in Fig. 1. Without any assumption on the properties of the graph signal, the three candidate graphs are all valid choices. However, if we assume that the signal is smooth on the underlying graph, which loosely means that neighboring vertices share similar signal values, then $G_1$ is obviously a more reasonable choice than $G_2$ and $G_3$. Our objective is exactly to learn a graph similar to $G_1$ when the signal is expected to be smooth on an “unknown” graph topology. This shows that learning a meaningful graph by enforcing desired signal properties on the observed signals, such as smoothness, is surely beneficial for the appropriate representation of the data. This is exactly the objective of our learning framework. Specifically, we consider in this paper the following problem:
Fig. 1. The same signal can potentially live on different graphs. In these figures, the red bars pointing upwards and the blue bars pointing downwards represent positive and negative signal values, respectively. The length of the bars reflects the magnitude of the values. In this illustration, while all the choices are valid a priori, we are looking for the graph that offers desirable properties of the resulting graph signal. Specifically, in this example, choosing graph $G_1$ favors the smoothness property of the graph signal.

**Graph learning for signal representation.** Given a set of observed data $X = \{x_i\}_{i=1}^p (x_i \in \mathbb{R}^n)$ that are considered as signals defined on a weighted and undirected graph $G$ of $n$ vertices, we would like to infer an optimal topology of $G$, namely, its edges and the associated weights, that results in given statistical properties of $X$, in particular the smoothness of the graph signals.

In this paper, we propose to define the relationships between signals and graphs by revisiting the representation learning theory [3] used in the traditional signal processing setting. Specifically, we consider a factor analysis model used in the theory of representation learning, where the Gaussian prior is a widely adopted probabilistic prior imposed on the latent variables that explain the observed signals. This is equivalent to imposing a Gaussian assumption on the observed signals. We generalize this Gaussian prior in the graph setting and impose it on the latent variables that control the observations, which are graph signals. The transformation from the latent variables to the observed signals now involves information about the topology of the graph. As a result, we can define joint properties (or a joint model) between the signals and the graph, such that the signal representation is consistent with the Gaussian prior. Specifically, we show that this generalization leads to a PCA-like representation for the graph signals, which turns out to be a smooth signal representation on graphs.

We then design an algorithm for graph learning that favors signal representations which are smooth and consistent with the statistical prior defined for the data. Specifically, given the input signal observations,
our algorithm iterates between the updates of the graph and the signals such that the variations of the signals on the learned graph are minimized upon convergence. We test our graph learning algorithm on several synthetic experiments, where we show that it efficiently infers the topology of the groundtruth graphs, and outperforms a learning algorithm for the estimation of a sparse inverse covariance matrix for Gaussian graphical models [4]. We further demonstrate the meaningfulness of the proposed framework through two practical examples. In the first example, we show that the proposed algorithm is able to infer the altitude relationships between 89 measuring stations in Switzerland from the average monthly temperature observations collected at these stations between 1981 and 2010. In the second example, we demonstrate that our algorithm is able to infer the political relationships between the 26 Swiss cantons from the votation data collected in the Swiss referendums for 37 federal popular initiatives between 2008 and 2012.

We provide two remarks here. First, the objective of our graph learning problem is to enforce desired properties on the training signals, so that the resulting graph signal representations are efficient for processing and analyzing the testing signals. This is slightly different from the objective of frameworks for learning Gaussian graphical models [5], [6] proposed in machine learning, where the learned graph usually represents the conditional independence structure between the random variables. Second, although we consider that the learned graph is mainly helpful for constructing efficient representations of signals, we expect it to be useful in graph-based learning tasks as well, such as clustering and classification. We will discuss both aspects in more details later on.

In summary, the proposed graph learning framework is one of the first rigorous frameworks to solve the challenging problem of graph learning in graph signal processing. It provides new insights into the understanding of the interactions between signals and graphs. It will also benefit numerous emerging real world applications, such as the analysis of transportation, biomedical, and social networks, where it is critical to infer hidden relationships between data entities.

The rest of the paper is organized as follows. In Section II we review the related work in the literature. In Section III we establish the link between signal representations in the traditional and graph settings when a Gaussian prior is imposed on the latent variables in a factor analysis model. We then propose our novel graph learning framework in Section IV and present the experimental results in Section V. We conclude the paper in Section VI.
II. RELATED WORK

Signal processing on graphs, as an emerging research field, has been attracting an increasing amount of interest from the signal processing community. Most of the research effort so far has been devoted to the efficient representation and processing of signals already defined on a given graph. Representative works include studies on the generalizations of the Fourier transform [7], [8], the wavelet transform [9], [10], [11], [12], [13], [14], dictionary learning methods [15], [16], and time-frequency analysis [17], [18] on graphs. However, most of the research effort in the signal processing community so far has been devoted to the representation and processing of the signals, while the graph is either given a priori, or chosen naturally from the application domain. Relatively less research effort has been devoted to the analysis and learning of the graph topologies. One of the most closely related works to ours has been proposed in [19], where a regression framework is proposed to learn a sparse graph topology based on a fitness metric between the signals and the graph. Another two examples are the works of [20] and [21], where the authors have proposed to use correlations between wavelet coefficients of the time-series of brain signals, and Principal Component Analysis, respectively, to estimate functional connectivities of distinct brain regions, which can be considered as the learning of an activity graph between the brain regions. These examples all focus on medical analysis applications. Finally, in [22] the authors have studied the link between the precision matrix in a Gaussian Markov Random Field and the graph Laplacian matrix. They however have not considered explicitly the structure of the graph Laplacian and have not proposed specifically an algorithm to learn a graph topology.

In the meantime, there is a large amount of work from the statistical machine learning community that aims at solving learning problems that are related to ours, but from a different perspective. For example, one topic in the research of learning graphical models is to estimate a full-rank inverse covariance matrix from the observed data for Gaussian graphical models [23], [5], [6], [24], [25], [26], especially in the case when the number of observations is smaller than the sample dimension and the sample covariance becomes singular. Another studied problem is to infer the graph structure for discrete Markov Random Fields [27]. It is known that in case of a Gaussian graphical model, there is an exact correspondence between the location of the non-zero entries in the precision matrix and the existence of partial correlations between the random variables [28]. In this case, a maximum-likelihood estimator turns out to be the solution to a log-determinant program. The estimated precision matrix is therefore considered to carry information about the partial correlations between the random variables. However, it is worth noting that the learned precision matrix, which is full-rank and usually has both positive and negative correlations
between the random variables, is not a valid graph Laplacian matrix. The overall process cannot thus be interpreted precisely as learning a graph topology. Another line of works is related to metric and kernel learning, whose objective is to infer similarities or dissimilarities between a set of entities. For example, in [29] the authors have proposed a general optimization framework for learning metrics via linear transformations, and analyzed the minimization of the log-determinant divergence subject to linear constraints as a special case. Given the statistical nature of these problems - in a graphical model the graph denotes the conditional independence structure between the random variables - most of the approaches tackle the problem either from a statistical point of view, e.g., with a maximum-likelihood estimation (MLE), or from an information-theoretic point of view, e.g., with the Bayesian information criterion (BIC). An essential difference with graph learning in graph signal processing is, however, that these approaches do not study the learning problem from the viewpoint of the processing of the graph signals. Specifically, they focus on the correlations between random variables, but do not pay much attention to the analysis of the characteristics and properties of signals defined on the graph, such as the smoothness of the data on the estimated graph. The latter is however the central and ultimate objective of this paper, such that the learned representation permits effective analysis of the graph data.

To summarize, the problem of learning graph topologies is an important area that has been relatively overlooked in the signal processing community. Although approaches for solving related problems exist in the statistical machine learning community, these approaches do not learn graph topologies that are linked to the desired properties of the observed signals. The proposed learning framework in the present paper is motivated by these observations, and is designed to enforce particular smoothness properties of the signals in their representations through the learned graph topology.

III. Factor Analysis Framework

As our objective is to learn graph topologies that enforce desired properties of the observed signals, we tackle the problem of graph learning from a signal representation perspective. We propose our learning framework by first making the connections between the signal representations in the traditional setting and the graph setting. Then we will use this analysis in the next section to design a novel framework for learning graph topologies.

A. Signal representation in the traditional setting

In our analysis, we consider the factor analysis [30], [31] model as our signal model, which is a generic linear statistical model that tries to explain observations of a given dimension with a potentially smaller
number of unobserved latent variables. Such latent variables can obey given probabilistic priors and lead
to effective signal representations in both the traditional and graph settings. Specifically, we consider the
following model:

\[ x = Wh + u_x + \epsilon, \] (1)

where \( x \in \mathbb{R}^n \) is the observation, \( h \in \mathbb{R}^k \) is the latent variable that controls \( x \), and \( W \in \mathbb{R}^{n \times k} \) is the
full rank representation matrix that linearly relates the two random variables. The parameter \( u_x \in \mathbb{R}^n \)
is the mean of \( x \), and \( \epsilon \) is a multivariate Gaussian noise that has mean zero and covariance \( \sigma_\epsilon^2 I_n \). Its
probability density function is given by:

\[ p(\epsilon) \sim \mathcal{N}(0, \sigma_\epsilon^2 I_n), \] (2)

where \( I_n \) represents the identity matrix of dimension \( n \). It is usually assumed that \( k < n \), so that the
latent variable \( h \) provides a more parsimonious signal representation than \( x \). Notice also that we do not
require the columns of \( W \) to be orthonormal. The conditional probability of \( x \) given \( h \) can then be written
as:

\[ p(x|h) \sim \mathcal{N}(Wh + u_x, \sigma_\epsilon^2 I_n). \] (3)

In the traditional setting, a Gaussian prior is conventionally imposed on the latent variable \( h \) in Eq. (3),
which generally leads to an efficient representation of the observation \( x \). Specifically, assume that \( h \)
follows a zero-mean multivariate Gaussian distribution with covariance matrix \( \sigma_h^2 I_n \):

\[ p(h) \sim \mathcal{N}(0, \sigma_h^2 I_n). \] (4)

Due to the properties of multivariate Gaussian distributions under linear transformations, \( x \) also follows
a multivariate Gaussian distribution:

\[ p(x) \sim \mathcal{N}(u_x, \sigma_h^2 WW^T + \sigma_\epsilon^2 I_n). \] (5)

Eq. (5) shows that the representation matrix \( W \) spans the same subspace as the \( k \) leading principal
components of the covariance matrix of \( x \). To show this more clearly, let us consider the following
eigendecomposition:

\[ \sigma_h^2 WW^T + \sigma_\epsilon^2 I_n = U\Lambda U^T, \] (6)

where \( U \) and \( \Lambda \) are the eigenvector and eigenvalue matrices of the covariance matrix. We can rewrite
Eq. (6) as follows:

\[ \sigma_h^2 WW^T = U\Lambda U^T - \sigma_\epsilon^2 I_n = U(\Lambda - \sigma_\epsilon^2 I_n)U^T. \] (7)
Since $W$ has rank $k$, Eq. (7) implies that the diagonal matrix $(\Lambda - \sigma^2 I_n)$ has only its first $k$ diagonal entries that are non-zero; their values are equal to $\sigma^2$. Therefore, Eq. (7) can be simplified as:

$$WW^T = U_k I_k U_k^T,$$

(8)

where $U_k \in \mathbb{R}^{n \times k}$ contains the $k$ leading principal components as columns. As a result, there must exist an orthogonal matrix $R \in \mathbb{R}^{k \times k}$ that satisfies

$$W = U_k R.$$

(9)

Eq. (9) implies that $W$ spans the same subspace as $U_k$. It has been pointed out in [3] that signal representation with the above factor analysis model provides a probabilistic interpretation of the highly successful representation learned by the PCA, which was originally presented in [32], [33]. The key observation here is that, under a Gaussian prior distribution of the latent variable $h$, the representation matrix $W$ in the factor analysis model spans the same subspace as the $k$ leading principal components of the covariance matrix of $x$. This provides an effective representation of the observation $x$ with a few latent variables. We will see in the following section that an analogous property exists in the graph setting; we will use this property in our graph learning framework.

Given the observation $x$ and the Gaussian prior distribution in Eq. (4), one is often interested in a maximum a posteriori (MAP) estimate of the latent variables $h$ that explain the observations $x$. In the following section, we will show an analogy of such an estimate in the graph setting, based on which we derive our graph learning framework.

**B. Signal representation in the graph setting**

We now consider the representation of signals in the graph setting, where the signals are defined on the vertex set of a graph. In the graph setting, for the representation of graph signals, we would like to generalize the factor analysis model in Eq. (1). The key is therefore to choose a representation matrix $W$ in Eq. (1) that relates the graph signals with the latent variable. Such a representation matrix should be defined in such a way that it reflects the topology of the graph. The Laplacian matrix of a graph $G$ provides a means to establish connections between signal representations in the traditional and graph settings. More specifically, the unnormalized (or combinatorial) graph Laplacian matrix $L$ is defined as:

$$L = D - W,$$

(10)

where $D$ is the degree matrix that contains the degrees of the vertices along the diagonal, and $W$ is the adjacency matrix of $G$. Since $L$ is a real and symmetric matrix, it has a complete set of orthonormal
eigenvectors and associated eigenvalues:

\[ L = \chi \Lambda \chi^T, \]  

(11)

where \( \chi \) is the eigenvector matrix, and \( \Lambda \) is the diagonal eigenvalue matrix where the eigenvalues are sorted in increasing order. The smallest eigenvalue is 0 with a multiplicity equal to the number of connected components of the graph \( G \), and all the other eigenvalues are strictly positive \[34\].

In our analysis, we propose to define the representation matrix as the eigenvector matrix \( \chi \) of the graph Laplacian \( L \). The motivation is that, under such a definition, a Gaussian prior imposed on the latent variable leads to a Gaussian assumption and smooth representations of the graph signals, as we will see later. One may also notice that, in Eq. \([1]\), the representation matrix \( W \) is a rank-\( k \) tall and thin matrix, while in the graph setting we choose to use \( \chi \) that is a rank-\( n \) square matrix. The motivation for this choice is that we would like to construct our model using all the eigenvectors of the graph Laplacian \( L \), not just the \( k \) leading ones.

Specifically, let us consider the following model:

\[ x = \chi h + u_x + \epsilon, \]  

(12)

where \( x \in \mathbb{R}^n \) represents the observed graph signal, \( h \in \mathbb{R}^n \) represents the latent variable that controls the graph signal \( x \) through the representation matrix \( \chi \), \( u_x \in \mathbb{R}^n \) is the mean of \( x \), and \( \epsilon \) is a multivariate Gaussian noise with mean zero and covariance \( \sigma^2 \epsilon I_n \). The probability density function of \( \epsilon \) is thus given by:

\[ p(\epsilon) \sim \mathcal{N}(0, \sigma^2 \epsilon I_n). \]  

(13)

Similarly to the traditional setting, we impose a Gaussian prior on the latent variable \( h \). Specifically, we assume that the latent variable \( h \) follows a degenerate zero-mean multivariate Gaussian distribution with covariance matrix \( \Lambda^\dagger \), which is the Moore-Penrose pseudoinverse of \( \Lambda \):

\[ p(h) \sim \mathcal{N}(0, \Lambda^\dagger). \]  

(14)

The conditional probability of \( x \) given \( h \), and the probability of \( x \), are respectively given as:

\[ p(x|h) \sim \mathcal{N}(\chi h + u_x, \sigma^2 \epsilon I_n), \]  

(15)

\[ p(x) \sim \mathcal{N}(u_x, L^\dagger + \sigma^2 \epsilon I_n), \]  

(16)

where we have used in Eq. \([16]\) the fact that the pseudoinverse of \( L \), \( L^\dagger \), admits the following eigendecomposition:

\[ L^\dagger = \chi \Lambda^\dagger \chi^T. \]  

(17)
It can be seen from Eq. (16) that, in a noise-free scenario where $\sigma_x = 0$, $x$ also follows a degenerate multivariate Gaussian distribution with zero-mean and covariance $L^\dagger$. In this case, $x$ can be seen as a Gaussian Markov Random Field (GMRF) with respect to the graph $G$, where the inverse covariance matrix, or the so-called precision matrix, is chosen to be the graph Laplacian $L$. Notice that the GMRF is a very generic model such that the precision matrix can be defined with much freedom, as long as its non-zero entries encode the partial correlations between the random variables, and as long as their locations correspond to the edges in the graph [28]. In the image analysis literature, the graph Laplacian $L$ is commonly adopted as the precision matrix of the GMRFs that model images with Gaussian distribution priors [22]. By defining the representation matrix in the factor analysis model as the eigenvector matrix $\chi$ and assuming that the latent variable follows a degenerate Gaussian distribution with covariance $\Lambda^\dagger$, we can recover a GMRF model with a precision matrix equal to $L$ in a noise-free scenario.

In the presence of noise, we see from Eq. (16) that, under a Gaussian prior on the latent variable $h$, the representation matrix $\chi$ is the eigenvector matrix of the covariance of $x$. Indeed, the covariance matrix $L^\dagger + \sigma_x^2 I_n$ admits the following eigendecomposition:

$$L^\dagger + \sigma_x^2 I_n = \chi(\Lambda^\dagger + \sigma_x^2 I_n)\chi^T. \tag{18}$$

This is analogous to the observation that we made in the traditional setting, where the representation matrix $W$ spans the same subspace as the $k$ leading principal components of the covariance of $x$ under a Gaussian prior on $h$. The representation in Eq. (12) can thus be considered as a PCA-like representation for the graph signal $x$. More importantly, it leads to smoothness properties for the signal on the graph, as we show below.

Recall that the latent variables $h$ explain the graph signal $x$ through the representation matrix $\chi$, namely, the eigenvector matrix of the graph Laplacian. Given the observation $x$ and the multivariate Gaussian prior distribution of $h$ in Eq. (14), we are thus interested in a MAP estimate of $h$. Specifically, by applying Bayes’ rule and assuming without loss of generality that $u_x = 0$, the MAP estimate of the latent variable $h$ can be written as follows [35]:

$$h_{\text{MAP}}(x) := \arg \max_h p(h|x)$$

$$= \arg \max_h p(x|h)p(h)$$

$$= \arg \min_h \left( -\log p_E(x - \chi h) - \log p_H(h) \right). \tag{19}$$

Now, from the probability distributions shown in Eq. (13) and Eq. (14), the above MAP estimate of
Eq. (19) can be expressed as:

\[ h_{\text{MAP}}(x) = \arg \min_h ||x - \chi h||_2^2 + \alpha h^T \Lambda h, \] (20)

where \( \alpha \) is some constant parameter. In a noise-free scenario where \( x = \chi h \), Eq. (20) corresponds to minimizing the following quantity:

\[ h^T \Lambda h = (\chi^T x)^T \Lambda \chi^T x = x^T \chi \Lambda \chi^T x = x^T L x. \] (21)

The Laplacian quadratic term in Eq. (21) is usually considered as a measure of smoothness of the signal \( x \) on \( G \) [36]. Therefore, we see that in a factor analysis model in Eq. (12), a Gaussian prior in Eq. (14) imposed on the latent variable \( h \) leads to smoothness properties for the graph signal. Similar observations can be made in a noisy scenario, where the main component of the signal \( x \), namely, \( \chi h \), is smooth on the graph. We are going to make use of the above observations in our graph learning algorithm in the following section.

IV. LEARNING GRAPHS UNDER SIGNAL SMOOTHNESS PRIOR

We have seen in the previous section that, given a Gaussian prior in the factor analysis model of the graph signals, the MAP estimate of \( h \) in Eq. (20) implies that the signal observations form smooth graph signals. Specifically, notice in Eq. (20) that both the representation matrix \( \chi \) and the inverse covariance matrix \( \Lambda \) of the Gaussian prior distribution imposed on \( h \) come from the graph Laplacian \( L \). They respectively represent the eigenvector and eigenvalue matrices of \( L \). When the graph is unknown, we can therefore have the following joint optimization problem of \( \chi, \Lambda \) and \( h \) in order to infer the graph topology:

\[ \min_{\chi,\Lambda,h} ||x - \chi h||_2^2 + \alpha h^T \Lambda h. \] (22)

Eq. (22) can be simplified with the change of variable \( y = \chi h \) to:

\[ \min_{L,y} ||x - y||_2^2 + \alpha y^T Ly. \] (23)

According to the factor analysis model in Eq. (12), \( y \) can be considered as a “noiseless” version of the zero-mean observation \( x \). Due to the properties of the graph Laplacian \( L \), the quadratic form \( y^T L y \) in Eq. (23) is usually considered as a measure of smoothness of the signal \( y \) on \( G \). Solving the problem of Eq. (23) is thus equivalent to finding jointly \( L \) (which is equivalent to the topology of the graph) and \( y \), such that \( y \) is close to the observation \( x \), and at the same time \( y \) is smooth on the learned graph \( G \). As a result, it enforces the smoothness property of the observed signals on the learned graph.
The optimization problem of Eq. (23) is ill-posed. We propose to solve this problem with the following objective function given in a matrix form:

$$\min_{L, Y} \|X - Y\|_F^2 + \alpha \text{tr}(Y^TLY) + \beta \|L\|_F^2,$$

subject to:

$$\text{tr}(L) = n,$$
$$L_{ij} = L_{ji} \leq 0, \ i \neq j,$$
$$L \cdot 1 = 0,$$

where $X \in \mathbb{R}^{n \times p}$ contains the $p$ input data samples $\{x_i\}_{i=1}^p$ as columns, $\alpha$ and $\beta$ are two regularization parameters that are positive, and 1 and 0 denote the constant one and zero vectors. The Frobenius norm of $L$ is added as a penalty term in the objective function to improve the numerical stability of the solution. Together with the Laplacian quadratic term, it will also influence the sparsity of the learned graph (as we will see in the experimental section\(^1\)). The first constraint (the trace constraint) in Eq. (24) permits to avoid trivial solutions, and the second and third constraints guarantee that the learned $L$ is a valid Laplacian matrix.

The optimization problem of Eq. (24) is not jointly convex in $L$ and $Y$. Therefore, we adopt an alternating optimization scheme where, at each step, we fix one variable and solve for the other variable. Specifically, at the first step, for a given $Y$, we solve the following optimization problem with respect to $L$:

$$\min_{L} \alpha \text{tr}(Y^TLY) + \beta \|L\|_F^2,$$

subject to:

$$\text{tr}(L) = n,$$
$$L_{ij} = L_{ji} \leq 0, \ i \neq j,$$
$$L \cdot 1 = 0.$$

At the second step, $L$ is fixed and we solve the following optimization problem with respect to $Y$:

$$\min_{Y} \|X - Y\|_F^2 + \alpha \text{tr}(Y^TLY).$$

Both the problems of Eq. (25) and Eq. (26) can be casted as convex optimization problems with unique solutions. First, the problem of Eq. (25) can be written as a quadratic program. In more details, notice that the matrix $L \in \mathbb{R}^{n \times n}$ in the problem of Eq. (25) is symmetric, which means that we only need

\(^1\)Notice that we do not impose a $L^1$-norm penalty on $L$ because we do not specifically assume that the learned graph is sparse.
to solve for the lower triangular part of $L$, that is, the $\frac{n(n+1)}{2}$ entries on and below the main diagonal. Therefore, instead of the square matrix form, we solve for the half-vectorization of $L$ that is obtained by vectorizing the lower triangular part of $L$. We denote the half-vectorization and vectorization of $L$ as $\text{vech}(L) \in \mathbb{R}^{\frac{n(n+1)}{2}}$ and $\text{vec}(L) \in \mathbb{R}^{n^2}$, respectively, and the former can be converted into the latter using the duplication matrix $\mathcal{M}_{\text{dup}}$:

$$\mathcal{M}_{\text{dup}} \text{vech}(L) = \text{vec}(L). \quad (27)$$

Now, by using Eq. (27) together with the fact that:

$$\text{tr}(Y^TLY) = \text{vec}(YY^T)^T \text{vec}(L), \quad (28)$$

and

$$||L||_F^2 = \text{vec}(L)^T \text{vec}(L), \quad (29)$$

we can rewrite the problem of Eq. (25) as:

$$\min_{\text{vech}(L)} \alpha \text{vec}(YY^T)^T \mathcal{M}_{\text{dup}} \text{vech}(L) + \beta \text{vech}(L)^T \mathcal{M}_{\text{dup}}^T \mathcal{M}_{\text{dup}} \text{vech}(L),$$

s.t. $A \text{vech}(L) = 0,

$$B \text{vech}(L) \leq 0,$$

(30)

where $A$ and $B$ are the matrices that handle the equality and inequality constraints in Eq. (25). The problem of Eq. (30) is a quadratic program with respect to the variable $\text{vech}(L)$ subject to linear constraints, and can be solved efficiently via interior point methods [37]. As we can see, the computational complexity scales quadratically with the number of vertices in the graph. With graphs of very large number of vertices, we can instead use operator splitting methods (e.g., alternating direction method of multipliers (ADMM) [4]) to find an approximate solution. Finally, once we solve the problem of Eq. (30), we convert $\text{vech}(L)$ into the square matrix form in order to solve the problem of Eq. (26).

Second, the problem of Eq. (26) has the following closed form solution:

$$Y = (I_n + \alpha L)^{-1}X. \quad (31)$$

In practice, since the matrix $I_n + \alpha L$ is Hermitian and positive-definite, we can use Cholesky factorization to compute $Y$ efficiently [37]. We then alternate between these two steps to get the final solution to the problem of Eq. (24), and we generally observe convergence within a few iterations. The complete algorithm is summarized in Algorithm 1.

We finally remark that the proposed learning framework has some similarity with the one in [19], where the authors have proposed a similar objective as the one in Eq. (24), based on a smoothness or
Algorithm 1 Graph Learning for Smooth Signal Representation (GL-SigRep)

1: **Input:** Input signal $X$, number of iterations $\text{iter}$, $\alpha$, $\beta$

2: **Output:** Output signal $Y$, graph Laplacian $L$

3: **Initialization:** $Y = X$

4: **for** $t = 1, 2, ..., \text{iter}$ **do**:

5: **Step to update Graph Laplacian $L$:**

6: Solve the optimization problem of Eq. (25) to update $L$.

7: **Step to update $Y$:**

8: Solve the optimization problem of Eq. (26) to update $Y$.

9: **end for**

10: $L = L^{\text{iter}}$, $Y = Y^{\text{iter}}$.

fitness metric of the signals on graphs. However, we rather take here a probabilistic approach that is analogous to the one in the traditional signal representation setting with the factor analysis model. This gives us an extra data fitting term $||X - Y||^2_F$ in the objective function of the optimization problem of Eq. (24). In practice, when the power of Laplacian is chosen to be 1, the problem in [19] corresponds to finding the solution to a single instance of the problem of Eq. (25) by assuming that $X = Y$.

V. EXPERIMENTS

In this section, we evaluate the performance of the proposed graph learning algorithm. We first describe the general experimental setting and the implementation details for the algorithms under comparison, and then we present the experimental results on synthetic and real world data.

A. Experimental settings

We test the performance of Algorithm 1 by comparing the graph learned from sets of synthetic or real world observations to the groundtruth graph. We provide visual and quantitative comparisons, where we compare the existence of edges in the learned graph to the ones of the groundtruth graph. We use four evaluation criteria commonly used in information retrieval [38] to test the performance of our algorithm:

- **Precision:** Precision is defined as:

  \[ \text{Precision} = \frac{TP}{TP + FP}, \]  

  \[
  (32)
  \]
where $TP$ and $FP$ are the numbers of true positives and false positives. It evaluates the percentage of correct edges in the learned graph, that are the edges that are also present in the groundtruth graph.

- **Recall**: Recall is defined as:
  \[
  \text{Recall} = \frac{TP}{TP + FN},
  \]
  where $FN$ is the number of false negatives. It evaluates the percentage of the edges in the groundtruth graph that are present in the learned graph.

- **F-measure**: F-measure is defined as the harmonic mean of Precision and Recall:
  \[
  \text{F-measure} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}.
  \]
  The F-measure takes into account both Precision and Recall to measure the accuracy of the obtained results.

- **Normalized Mutual Information (NMI)**: NMI is defined as:
  \[
  \text{NMI}(\Omega, C) = \frac{I(\Omega; C)}{[H(\Omega) + H(C)]/2},
  \]
  where $I(\Omega; C)$ is the mutual information between clusters $\Omega$ and classes $C$, and $H(\Omega)$ and $H(C)$ represent the entropies of the clusters and classes, respectively. The NMI measures the mutual dependence between the obtained results and the groundtruth. In our experiments, we first compute a 2-cluster partition of all the vertex pairs using the learned graph, based on whether or not there exists an edge between every pair of vertices. To compute the NMI, we then compare this partition with the 2-class partition obtained in the same way using the groundtruth graph.

In our experiments, we solve the optimization of Eq. (25) using the convex optimization package CVX [39], [40]. The algorithm stops when the maximum number of iterations is reached or the absolute change in the objective is smaller than $10^{-4}$. In most of the cases, we observed that the algorithm converges within a few iterations. The experiments are carried out on different sets of parameters, namely, for different values of $\alpha$ and $\beta$ in Eq. (24). Finally, we prune the insignificant edges that have a weight smaller than $10^{-4}$ in the learned graph.

In the absence of other methods for learning graph topologies that enforce smoothness of graph signal representations in the literature, we compare the proposed graph learning framework to a state-of-the-art approach for estimating a sparse inverse covariance matrix for GMRFs. Specifically, the works in [5], [6] propose to solve the following $L^1$-regularized log-determinant program:

\[
\min_{L_{pre}} \text{tr}(SL_{pre}) - \log \det(L_{pre}) + \lambda\|L_{pre}\|_1,
\]
where $L_{pre}$ is the precision (inverse covariance) matrix to estimate, $S = XX^T$ is the sample covariance matrix, $\lambda$ is a regularization parameter, $\det(\cdot)$ denotes the determinant, and $\| \cdot \|_1$ denotes the $L^1$-norm. The main motivation behind the optimization of Eq. (36) is as follows. When the number of samples is smaller than the sample dimension, the sample covariance $S$ is not full rank. In this case, the optimization of Eq. (36) tries to find a full rank precision matrix $L_{pre}$ (which is enforced by the log-determinant term), that is close to the sample covariance $S$ (which is enforced by the trace term). Additionally, the $L^1$ penalty is introduced to enforce the sparsity of $L_{pre}$ in terms of the $L^1$-norm. The problem of Eq. (36) is conceptually similar to the problem of Eq. (24), in the sense that both can be interpreted as estimating the precision matrix of a GMRF. An important difference is however that the precision matrix in our framework is a valid graph Laplacian, while the one in Eq. (36) is not. Indeed, $L_{pre}$ is a full rank matrix that usually have both positive and negative off-diagonal entries, reflecting both positive and negative correlations. Therefore, $L_{pre}$ cannot be interpreted as a graph topology for defining graph signals; it rather only reflects the partial correlations between the random variables that control the observations. This major difference leads to different performances of the two approaches. In our experiments, we solve the $L^1$-regularized log-determinant program of Eq. (36) with the ADMM [4]. We denote this algorithm as GL-LogDet. Even if the comparison with GL-LogDet is not completely fair since it does not have the constraint to form a valid Laplacian, it is nevertheless interesting since the learned precision matrix also encodes the information about the partial correlations between the variables. In our experiments, following the suggestions in [4], we set the augmented Lagrangian parameter and the over-relaxation parameter in GL-LogDet to be 1 and 1.5, respectively. We test GL-LogDet based on different choices of the parameter $\lambda$ in Eq. (36). In the evaluation, all the off-diagonal non-zero entries whose absolute values are above the threshold of $10^{-4}$ are considered as valid correlations\(^2\). These correlations are then considered as learned “edges” and compared against the edges in the groundtruth graph for performance evaluation.

B. Results on synthetic data

We first carry out experiments on three different synthetic graphs of 20 vertices, namely, a graph whose edges are determined based on Euclidean distances between vertices, and two graphs that follow the Erdős-Rényi model [41] and the Barabási-Albert model [42], respectively. For the first graph, we

\(^2\)Since the precision matrix learned by GL-LogDet seldom contains entries with values smaller than $10^{-4}$, the thresholding does not make a difference in the evaluation. Nevertheless, we still implement this step for a fair comparison.
generate the coordinates of the vertices uniformly at random in the unit square, and compute the edge weights between every pair of vertices using the Euclidean distances between them and a Gaussian radial basis function (RBF): \( \exp \left( -d(i,j)^2 / 2\sigma^2 \right) \), with the width parameter \( \sigma = 0.5 \). We then remove all the edges whose weights are smaller than 0.75. Next, we use the Erdős-Rényi (ER) model with edge probability 0.2 to generate a random graph, that is, each possible edge is included in the graph with probability 0.2 independently of other edges. Finally, we use the Barabási-Albert (BA) model to generate a scale-free random graph. Specifically, the BA graph in our experiments is generated by adding one new vertex to the graph at each time, connecting to one existing vertex in the graph. The probability of the new vertex attaching to a given existing vertex in the graph is proportional to the ratio of the degree of that existing vertex to the sum of degrees of all the existing vertices. The BA and ER graphs are important random graph models studied in network science. Specifically, the former have power-law (or scale-free) degree distributions similarly to many networks observed in real world, while the latter do not. The BA and ER graphs in our experiments have unitary edge weights.

Given a groundtruth synthetic graph, we compute the graph Laplacian \( L \) and normalize the trace according to Eq. (24). Then, for each graph, we generate 100 signals \( X = \{x_i\}_{i=1}^{100} \) that follow the distribution shown in Eq. (16) with \( \mu_x = 0 \) and \( \sigma_\epsilon = 0.5 \). We then apply GL-SigRep and GL-LogDet to learn the graph Laplacian or the precision matrix, respectively, given only the signals \( X \).

We first show visually the learned graph Laplacians or precision matrices for the algorithms GL-SigRep and GL-LogDet. In Fig. 2, we show from the top to the bottom rows the Laplacian matrices of the groundtruth graph, the graph Laplacians learned by GL-SigRep, the precision matrices learned by GL-LogDet, and the sample covariance matrices \( S = XX^T \), for one random instance of each of the three graph models\(^3\). First, we see that, although the positions of the large entries in the sample covariance matrices partially correspond to the positions of the edges in the groundtruth graph, they generally contain many noisy correlation values. Next, for all the three types of graphs, the graph Laplacian matrices learned by GL-SigRep is visually more consistent with the groundtruth data than the precision matrices learned by GL-LogDet. In addition to the position of the edges, the edge weights in the Laplacians learned by GL-SigRep is similar to the ones in the groundtruth graph, while the correlations in the precision matrices learned by GL-LogDet are relatively small (this is especially the case for the BA graph). Possible

\(^3\)These results are obtained based on the parameters, namely, \( \alpha \) and \( \beta \) in GL-SigRep and \( \lambda \) in GL-LogDet, that lead to a similar number of edges as the ones in the groundtruth graph. More discussion about the choices of these parameters are presented later. The values of the sample covariance matrices are scaled before the visualization.
Fig. 2. The learned graph Laplacian or precision matrices. From the top to the bottom rows are the groundtruth Laplacians, the Laplacians learned by GL-SigRep, the precision matrices learned by GL-LogDet, and the sample covariances. From the left to the right columns we show the Gaussian RBF graph, the ER graph, and the BA graph.
TABLE I

Graph learning performance for GL-SigRep and GL-LogDet.

(a) The Gaussian RBF graph.

| Algorithm    | F-measure | Precision | Recall | NMI  |
|--------------|-----------|-----------|--------|------|
| GL-SigRep    | 0.8803    | 0.8535    | 0.9108 | 0.5902 |
| GL-LogDet    | 0.4379    | 0.2918    | 0.8851 | 0.0220 |

(b) The ER graph.

| Algorithm    | F-measure | Precision | Recall | NMI  |
|--------------|-----------|-----------|--------|------|
| GL-SigRep    | 0.7495    | 0.7185    | 0.7901 | 0.4126 |
| GL-LogDet    | 0.3181    | 0.2235    | 0.5587 | 0.0109 |

(c) The BA graph.

| Algorithm    | F-measure | Precision | Recall | NMI  |
|--------------|-----------|-----------|--------|------|
| GL-SigRep    | 0.8898    | 0.9398    | 0.8474 | 0.7345 |
| GL-LogDet    | 0.3277    | 0.2657    | 0.5000 | 0.0665 |

Explanations of this observation are that: (i) GL-LogDet estimates a full rank precision matrix instead of the groundtruth graph Laplacian, and (ii) the \( L^1 \)-norm penalty in the optimization of Eq. (36) leads to many relatively small correlation values.

Next, we quantitatively evaluate the performances of the two graph learning algorithms in recovering the positions of the edges in the groundtruth. In Table I, we show the best F-measure, Precision, Recall and NMI scores achieved by the two algorithms averaged over ten random instances of the three graphs with the associated signals \( X \). Our algorithm GL-SigRep clearly outperforms GL-LogDet in terms of all the evaluation criteria. Especially, for the Gaussian RBF and BA graphs, GL-SigRep achieves an average F-measure score close to 0.9, which means that the learned graphs have topologies that are very similar to the groundtruth ones.

To better understand the behavior of GL-SigRep under different sets of parameters, we plot in Fig. 3 the numbers of edges in the learned graph, and the F-measure scores, under 121 different combinations of the parameters \( \alpha \) and \( \beta \) in Eq. (24), for a random instance of the Gaussian RBF graph. First, we see that the number of edges in the learned graph decreases as \( \beta \) decreases and \( \alpha \) increases. The intuitions behind this behavior are as follows. When \( \beta \) increases, the Frobenius norm of \( L \) in the objective function in Eq. (24) tends to be small, leading to a more uniform distribution of the entries, so that the number of edges tends to increase. Decreasing \( \beta \) leads to the opposite effect. When \( \alpha \) increases, the trace of the
Fig. 3. (a) The number of edges in the learned graph, and (b) the F-measure score, under 121 different combinations of the parameters $\alpha$ and $\beta$ for an instance of the Gaussian RBF graph.

The quadratic term tends to be small. The algorithm thus favors a smaller number of non-zero entries in $L$, and the number of edges decreases. Therefore, both parameters $\alpha$ and $\beta$ implicitly affect the sparsity of the learned graph Laplacian. More interestingly, Fig. 3(a) and Fig. 3(b) show that both the number of edges and the F-measure scores are similar for the values of $\alpha$ and $\beta$ that lead to the same ratio $\frac{\beta}{\alpha}$. This suggests that the trace of the quadratic term and the Frobenius norm are the dominating factors in the optimization of Eq. (24), rather than the data fidelity term. This implies that, in practice, we may only need to search for an appropriate ratio $\frac{\beta}{\alpha}$ to maximize the learning performance of the algorithm.

Next, we show in Fig. 4(a) and Fig. 5(a) the number of edges in the graphs learned by GL-SigRep and the learning performances evaluated based on the four criteria, respectively, for the same Gaussian RBF graph as before under different ratios of $\beta$ to $\alpha$. As expected, the number of edges decreases as the ratio of $\beta$ to $\alpha$ decreases. Looking at Fig. 4(a) and Fig. 5(a) together, we see that, as the number of edges approaches the number of edges in the groundtruth graph (in this case, 56 edges), the Recall stays high and the Precision increases rapidly, which makes the F-measure increase. When the number of edges in the learned graph is close to the one in the groundtruth graph, the curves for the Precision and the Recall intersect and the F-measure reaches its peak. After that, although the Precision keeps increasing towards 1, the Recall drops rapidly as fewer and fewer edges are detected, leading to a decreasing trend in the F-measure. A similar trend can be observed in the curve for the NMI score. These together show that GL-SigRep is able to learn a graph that is very close to the groundtruth graph when the number of edges matches the number of edges in the groundtruth graph. For comparison, we plot the same curves in
Fig. 4. (a) Number of edges in the graphs learned by GL-SigRep for different ratios $\frac{\beta}{\alpha}$. (b) Number of correlations in the precision matrices learned by GL-LogDet for different values of $\lambda$.

Fig. 5. Performance of (a) GL-SigRep for different ratios $\frac{\beta}{\alpha}$, and (b) GL-LogDet for different values of $\lambda$.

Fig. 4(b) and Fig. 5(b) for GL-LogDet, under different values of the parameter $\lambda$ in Eq. (36). In this case, GL-LogDet learns an increasing number of correlations, many of which with relatively small values, as $\lambda$ decreases. The Precision and Recall curves still intersect at a point where the number of correlations matches most closely the number of edges in the groundtruth graph. However, both scores are rather low at this point, which leads to an unsatisfactory F-measure score. Similarly, the NMI scores are very low. Overall, these comparisons show that GL-SigRep achieves better performance than GL-LogDet in learning the positions of the edges.

Finally, we investigate the influence of the number of signals available for learning, and the level of
noise present in the data. In Fig. 6 we show the performance of the two algorithms for different numbers of signals for a given instance of the Gaussian RBF graph. As we can see in Fig. 6 the performance of both algorithms increases as more signals are available to learn the graph Laplacian and the precision matrix, respectively. The benefit of having more signals seems slightly larger for GL-SigRep, as the F-measure increases quickly when more than 20 signals are available. In Fig. 7 we show the performance of the algorithms for different values of the standard deviation of the Gaussian noise $\sigma_\epsilon$, for the same groundtruth graph. We see that the performance of both algorithms is rather stable when the noise intensity increases, although GL-SigRep seems to be more sensitive to noise in this experiment where the F-measure drops only slightly.
C. Learning meteorological graph from temperature data

We now test the proposed graph learning framework on real world data. Specifically, we consider the average monthly temperature data collected at 89 measuring stations in Switzerland (shown in Fig. 8(a)) during the period between 1981 and 2010. This leads to 12 signals (i.e., one per month), each of dimension 89, which correspond to the average temperatures at each of the measuring stations. By applying the proposed graph learning algorithm, we would like to infer a graph where stations with similar temperature evolutions across the year are connected. In other words, we aim at learning a graph on which the observed temperature signals are smooth. In this case, the natural choice of a geographical graph based on physical distances between the stations does not seem appropriate for representing the similarity of temperature values between these stations. Indeed, Fig. 8(b)-(d) shows the average temperatures in Switzerland in February, June and October, and we can see that the evolution of temperatures at most of the stations follow very similar trends and are thus highly correlated, regardless of the geographical distances between them. On the other hand, it turns out that altitude is a more reliable source of information to determine temperature evolutions. For instance, as we observed from the data, the temperature at two stations, Jungfraujoch and Piz Corvatsch, follow similar trends that are clearly different from other stations, possibly due to their similar altitudes (both are more than 3000 metres above sea level). Therefore, we build a groundtruth graph that reflects the similarity between stations in terms of their altitudes. More specifically, we connect two stations with an unitary weight if and only if their altitude difference is smaller than 300 metres. The goal of our learning experiments is then to recover this altitude graph given the observed temperature signals.

We first show visual comparisons between the Laplacian of the groundtruth altitude-based graph, the graph Laplacian learned by GL-SigRep, the precision matrix learned by GL-LogDet, and the sample covariance matrix. For a more clear visualization, we focus on the top left part of the four matrices and plot them in Fig. 9. First, we see that the sample covariance does not match well the groundtruth, since it contains many correlation values that do not appear in the groundtruth graph, as it was the case for the synthetic data. This is especially due to a small number of samples (12) compared to the sample dimension (89) in this case. The comparisons between GL-SigRep and GL-LogDet show that the edges in the graph learned by our algorithm is again more consistent with the groundtruth data in terms of both positions and weights. Similarly to the case of synthetic experiments, we compare the number of edges in the learned graph and the $F$-measure scores for different ratios $\frac{B}{A}$ for GL-SigRep; the results are presented in Fig. 10(a) and Fig. 11(a). Clearly, we can observe the same behavior as in the synthetic
Fig. 8. (a) The locations of 89 measuring stations in Switzerland. (b)-(d) Average monthly temperature in February, June and October, respectively, in Switzerland during the period from 1981 to 2010. The color code in (b)-(d) represents the temperature in Celsius scale.

experiments, namely, when the number of edges matches closely the one in the groundtruth graph (i.e., 1169 edges), the curves for the Precision and the Recall intersect, and the F-measure and NMI scores are near their peaks. For the best parameter values, GL-SigRep achieves a F-measure close to 0.85, which indicates a very good recovery of the topology of the groundtruth graph. In Fig. 10(b) and Fig. 11(b), the same results are presented for GL-LogDet. We can see that the performance is less convincing in this case. Specifically, both the Precision and the Recall stay relatively low and do not come close to 1. As a result, the curve for F-measure is less satisfactory and a similar trend is observed for the NMI curve. It is worth noting that, in Fig. 10(b), the number of correlation values in the learned precision
The results for GL-SigRep and GL-LogDet are obtained based on the parameters $\alpha$, $\beta$ and $\lambda$ that lead to the best $F$-measure scores. The values of the sample covariance matrices are scaled before the visualization.

Fig. 9. Visual comparisons between (a) The groundtruth graph Laplacian. (b) The graph Laplacian learned by GL-SigRep. (c) The precision matrix learned by GL-LogDet. (d) The sample covariance matrix. The results for GL-SigRep and GL-LogDet are obtained based on the parameters $\alpha$, $\beta$ and $\lambda$ that lead to the best $F$-measure scores. The values of the sample covariance matrices are scaled before the visualization.

Matrices decreases as $\lambda$ decreases, before increasing again in the end. This is not consistent with the monotonically increasing trend that we have observed in Fig. 4(b) in the synthetic experiments. By taking a closer look at the precision matrices learned in both cases, we observe that the $L^1$-norm of the precision matrices always decreases as $\lambda$ increases. However, the decrease in the $L^1$-norm does not necessarily translate into a larger sparsity in terms of the number of non-zero entries. In practice, it could happen that more and more non-zero entries with smaller values appear in the precision matrices as the $L^1$-norm decreases, which is the case in Fig. 10(b). Overall, GL-SigRep is clearly more reliable than GL-LogDet in achieving a desired sparsity level for the learned graph.
In the example shown above, since the learned graph encodes relationships between temperature measurements at different measuring stations, we would like to separate these stations into disjoint clusters based on the learned graph such that different clusters correspond to different characteristics of the measure stations. In our experiment, we apply the spectral clustering algorithm proposed in [44] to the learned graph to partition the vertex set into two disjoint clusters. The results are shown in Fig. 12, where the red and blue dots represent two different clusters of stations. As we can see, the stations in the red cluster are mainly those built on the mountains, such as those in the Jura Mountains and Alps, while the ones in the blue cluster are mainly stations in flat regions. It is especially interesting to notice
Fig. 12. Two clusters of the measuring stations obtained by applying spectral clustering to the learned graph. The red cluster includes stations at higher altitudes and the blue cluster includes stations at lower altitudes.

that, the blue stations in the Alps region (from centre to the bottom right of the map) mainly lie in the valleys along main roads (such as those in the canton of Valais) or in the Lugano region. This shows that the obtained clusters capture the altitude information of the measuring stations hence confirms the quality of the learned graph topology.

D. Learning political graph from votation data

We now move onto the second real world example, where we consider votation data from the national referendums for 37 federal popular initiatives in Switzerland between 2008 and 2012 [45]. Specifically, we consider the percentage of votes supporting each initiative in the 26 Swiss cantons as our observed signal. This leads to 37 signals, each of dimension 26. By applying the proposed graph learning framework, we would like to infer a graph that captures the political relationships between the Swiss cantons in terms of their votes in the national referendums.

Compared to the previous example, it is rather unclear what would be the groundtruth graph that captures the political relationships between the Swiss cantons. Therefore, we evaluate the performance of our learning algorithm as follows. We apply the proposed algorithm GL-SigRep to learn a graph topology, and then partition the vertices of the graph into several disjoint clusters using the spectral clustering algorithm in [44]. That distributes the 26 cantons into several clusters based on their votes in the national referendums. We then evaluate the quality of the learned graph by interpreting the clustering results.
In Fig. 13, we show the two most stable 3-cluster partitions obtained by applying spectral clustering to the graphs learned by **GL-SigRep** under different parameters $\alpha$ and $\beta$. We can see that the blue cluster contains all the French-speaking cantons, while the yellow clusters contain most of the German-speaking cantons and the Italian-speaking canton Ticino. Then, the five cantons in the red cluster, namely, Uri, Schwyz, Nidwalden, Obwalden and Appenzell Innerrhoden, the first four of which constituting the so-called “primitive” cantons at the origin of Switzerland, are all considered among the most conservative cantons in Switzerland. Interestingly, the only difference between the two clusterings shown in Fig. 13 is the cluster membership of the canton Basel-Stadt. This can be explained by the observation that Basel regularly agrees with the cantons in the French-speaking part of Switzerland on referendums for close relations with the European Union. Therefore, these clustering results demonstrate that the graph learned by **GL-SigRep** indeed captures the hidden political relationships between the 26 Swiss cantons, which are consistent with the general understanding of their voting behaviors in the national referendums.

Finally, to confirm the meaningfulness of the clusterings shown in Fig. 13, we illustrate in Fig. 14 a 3-cluster partition of the cantons based on votation statistics in a recent national referendum for the initiative “Against mass immigration”. This referendum, which was held in February 2014, has the largest turnout of 55.8% in recent years. In Fig. 14 the cantons in the blue cluster voted against the initiative, while the ones in the yellow and red clusters voted for it. In particular, the seven cantons where the percentage of supporting the initiative is greater than 58.2% are grouped in the red cluster. As we can see, this partition is largely consistent with those in Fig. 13 with the only exceptions of the cantons of Zürich and Zug agreeing with the French-speaking cantons with small margins, where all the five cantons in the red cluster in Fig. 13 are among the seven that are most strongly against mass immigration. This
VI. CONCLUSION

In this paper, we have presented a framework for learning graph topologies from the signal observations under the assumption that the signals are smooth on the learned graphs. We have developed a method for learning graphs that enforce the smoothness property of the graph signals, under a Gaussian prior distribution imposed on the latent variables in a factor analysis model. In both synthetic and real world experiments, we have shown that the proposed method outperforms an algorithm for estimating a sparse inverse covariance matrix for Gaussian graphical models. The proposed graph learning framework opens new perspectives in the field of signal processing on graphs, and would surely benefit numerous emerging applications that involve analysis or processing of structured data.

ACKNOWLEDGEMENT

The authors would like to thank Prof. Antonio Ortega and Prof. Michael Bronstein for their insightful comments about the draft, and Mr. Chenhui Hu and Mr. Vassilis Kalofolias for their helpful discussions about the optimization.

REFERENCES

[1] D. I Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, “The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains,” IEEE Signal Processing Magazine, vol. 30, no. 3, pp. 83–98, May 2013.
[2] A. Sandryhaila and J. M. F. Moura, “Discrete signal processing on graphs,” IEEE Transactions on Signal Processing, vol. 61, no. 7, pp. 1644–1656, Apr 2013.
[3] Y. Bengio, A. Courville, and P. Vincent, “Representation learning: A review and new perspectives,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 35, no. 8, pp. 1798–1828, Aug 2013.
[4] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” Foundations and Trends in Machine Learning, vol. 3, no. 1, pp. 1–122, 2011.
[5] 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, vol. 9, pp. 485–516, Jun 2008.
[6] J. Friedman, T. Hastie, and R. Tibshirani, “Sparse inverse covariance estimation with the graphical lasso,” Biostatistics, vol. 9, no. 3, pp. 432–441, Jul 2008.
[7] D. I Shuman, B. Ricaud, and P. Vandergheynst, “A windowed graph Fourier transform,” in Proceedings of the IEEE Statistical Signal Processing Workshop (SSP), 2012.
[8] X. Zhu and M. Rabbat, “Approximating signals supported on graphs,” in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012.
[9] R. R. Coifman and M. Maggioni, “Diffusion wavelets,” Applied and Computational Harmonic Analysis, vol. 21, no. 1, pp. 53–94, Jul 2006.
[10] M. Gavish, B. Nadler, and R. Coifman, “Multiscale wavelets on trees, graphs and high dimensional data: Theory and applications to semi supervised learning,” in Proceedings of the International Conference on Machine Learning (ICML), 2010.
[11] D. K. Hammond, P. Vandergheynst, and R. Gribonval, “Wavelets on graphs via spectral graph theory,” Applied and Computational Harmonic Analysis, vol. 30, no. 2, pp. 129–150, Mar 2011.
[12] I. Ram, M. Elad, and I. Cohen, “Generalized tree-based wavelet transform,” IEEE Transactions on Signal Processing, vol. 59, no. 9, pp. 4199–4209, Sep 2011.
[13] S. K. Narang and A. Ortega, “Perfect reconstruction two-channel wavelet filter-banks for graph structured data,” IEEE Transactions on Signal Processing, vol. 60, no. 6, pp. 2786–2799, Jun 2012.
[14] D. I Shuman, C. Wiesmeyr, N. Holighaus, and P. Vandergheynst, “Spectrum-adapted tight graph wavelet and vertex-frequency frames,” in arXiv:1311.0897, 2013.
[15] X. Zhang, X. Dong, and P. Frossard, “Learning of structured graph dictionaries,” in Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2012.
[16] D. Thanou, D. I Shuman, and P. Frossard, “Learning parametric dictionaries for signals on graphs,” in IEEE Transactions on Signal Processing, to appear.
[17] D. I Shuman, B. Ricaud, and P. Vandergheynst, “Vertex-frequency analysis on graphs,” in arXiv:1307.5708, 2013.
[18] A. Agaskar and Y. M. Lu, “A spectral graph uncertainty principle,” IEEE Transactions on Information Theory, vol. 59, no. 7, pp. 4338–4356, Jul 2013.
[19] C. Hu, L. Cheng, J. Sepulcre, G. E. Fakhri, Y. M. Lu, and Q. Li, “A graph theoretical regression model for brain connectivity learning of Alzheimer’s disease,” in Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI), 2013.
[20] N. Leonardi and D. Van De Ville, “Wavelet frames on graphs defined by fMRI functional connectivity,” in Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI), 2011, pp. 2136–2139.
[21] N. Leonardi, J. Richiardi, M. Gschwind, S. Simioni, J.-M. Annoni, M. Schluep, P. Vuilleumier, and D. Van De Ville,
“Principal components of functional connectivity: A new approach to study dynamic brain connectivity during rest,” *NeuroImage*, vol. 83, pp. 937–950, Dec 2013.

[22] C. Zhang and D. Florencio, “Analyzing the optimality of predictive transform coding using graph-based models,” *IEEE Signal Processing Letters*, vol. 20, no. 1, pp. 106–109, Jan 2013.

[23] M. Yuan and Y. Lin, “Model selection and estimation in the Gaussian graphical model,” *Biometrika*, vol. 94, no. 1, pp. 19–35, 2007.

[24] A. J. Rothman, P. J. Bickel, E. Levina, and J. Zhu, “Sparse permutation invariant covariance estimation,” *Electronic Journal of Statistics*, vol. 2, pp. 494–515, 2008.

[25] P. Ravikumar, G. Raskutti, M. J. Wainwright, and B. Yu, “Model selection in Gaussian graphical models: High-dimensional consistency of ℓ₁-regularized MLE,” in *Advances in Neural Information Processing Systems 21 (NIPS)*, 2008, pp. 1329–1336.

[26] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, and P. Ravikumar, “Sparse inverse covariance matrix estimation using quadratic approximation,” in *Advances in Neural Information Processing Systems 24 (NIPS)*, 2011, pp. 2330–2338.

[27] P. Loh and M. J. Wainwright, “Structure estimation for discrete graphical models: Generalized covariance matrices and their inverses,” *The Annals of Statistics*, vol. 41, no. 6, pp. 3022–3049, 2013.

[28] H. Rue and L. Held, “Gaussian Markov random fields: Theory and applications,” *Chapman and Hall/CRC*, 2005.

[29] P. Jain, B. Kulis, J. V. Davis, and I. S. Dhillon, “Metric and kernel learning using a linear transformation,” *Journal of Machine Learning Research*, vol. 13, pp. 519–547, Mar 2012.

[30] D. J. Bartholomew, M. Knott, and I. Moustaki, “Latent variable models and factor analysis: A unified approach, 3rd Edition,” Wiley, Jul 2011.

[31] A. Basilevsky, “Statistical factor analysis and related methods,” *Wiley*, Jan 1994.

[32] S. Roweis, “EM algorithms for PCA and sensible PCA,” *CNS Technical Report CNS-TR-97-02, Caltech*, 1997.

[33] M. E. Tipping and C. M. Bishop, “Probabilistic principal component analysis,” *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, vol. 61, no. 3, pp. 611–622, 1999.

[34] F. R. K. Chung, “Spectral graph theory,” *American Mathematical Society*, 1997.

[35] R. Giribonval, “Should penalized least squares regression be interpreted as maximum a posteriori estimation?” *IEEE Transactions on Signal Processing*, vol. 59, no. 5, pp. 2405–2410, May 2011.

[36] D. Zhou and B. Schölkopf, “A regularization framework for learning from graph data,” in *ICML Workshop on Statistical Relational Learning*, 2004, pp. 132–137.

[37] S. Boyd and L. Vandenberghe, “Convex optimization,” *Cambridge University Press*, 2004.

[38] C. D. Manning, P. Raghavan, and H. Schütze, “Introduction to information retrieval,” *Cambridge University Press*, 2008.

[39] M. Grant and S. Boyd, “CVX: Matlab software for disciplined convex programming, version 2.0 beta,” [http://cvxr.com/cvx](http://cvxr.com/cvx), September 2013.

[40] ——, “Graph implementations for nonsmooth convex programs,” in *Recent Advances in Learning and Control*, ser. Lecture Notes in Control and Information Sciences, V. Blondel, S. Boyd, and H. Kimura, Eds. Springer-Verlag Limited, 2008, pp. 95–110, [http://stanford.edu/~boyd/graph_dcp.html](http://stanford.edu/~boyd/graph_dcp.html).

[41] P. Erdős and A. Rényi, “On the evolution of random graphs,” *Publications of the Mathematical Institute of the Hungarian Academy of Sciences*, vol. 5, pp. 17–61, 1960.

[42] A.-L. Barabási and R. Albert, “Emergence of scaling in random networks,” *Science*, vol. 286, no. 5439, pp. 509–512, Oct 1999.
[43] http://www.meteosuisse.admin.ch/web/en/climate/swiss_climate/tabellen.html

[44] A. Ng, M. Jordan, and Y. Weiss, “On spectral clustering: Analysis and an algorithm,” in Advances in Neural Information Processing Systems 14 (NIPS), 2001, pp. 849–856.

[45] http://www.swissvotes.ch/db/votes/search