LEARNING SPARSE GRAPHS UNDER SMOOTHNESS PRIOR

Sundeep Prabhakar Chepuri[†], Sijia Liu[‡], Geert Leus[†], and Alfred O. Hero III[‡]

[†]Delft University of Technology (TU Delft), The Netherlands
[‡]Department of EECS, University of Michigan, Ann Arbor, MI 48109, USA
Email: [†]{s.p.chepuri; g.j.t.leus}@tudelft.nl, [‡]{lsjxjtu, hero}@umich.edu.

ABSTRACT

In this paper, we are interested in learning the underlying graph structure behind training data. Solving this basic problem is essential to carry out any graph signal processing or machine learning task. To realize this, we assume that the data is smooth with respect to the graph topology, and we parameterize the graph topology using an edge sampling function. That is, the graph Laplacian is expressed in terms of a sparse edge selection vector, which provides an explicit handle to control the sparsity level of the graph. We solve the sparse graph learning problem given some training data in both the noiseless and noisy settings. Given the true smooth data, the posed sparse graph learning problem can be solved optimally and is based on simple rank ordering. Given the noisy data, we show that the joint sparse graph learning and denoising problem can be simplified to designing only the sparse edge selection vector, which can be solved using convex optimization.

Index Terms— Graph Learning, graph signal processing, graph sparsification, topology inference, sparse sampling.

1. INTRODUCTION

Graphs offer a way to describe and explain relationships in complex datasets, a central entity of modern data analysis, where data deluge is prominent [1–3]. In particular, the nodes of the graph denote the entities and the edges encode the pairwise relationship between these entities. Such entities are referred to as *graph signals*. Examples of such complex-structured data beyond traditional time-series include data residing on brain networks, gene networks, social networks, recommendation systems, transportation networks, and so on.

Having a good quality graph is central to any graph signal processing or machine learning task. In this paper, we are interested in the problem of learning the hidden graph topology behind the data. Due to the sheer quantity of data, we are motivated to select the simplest graphical models that adequately explain the data. In particular, we are interested in learning a sparse graph, i.e., a graph with a limited number of edges that adequately explains the input (or training) data. To realize this, we make a simple, but widely used assumption [1,4] that the data is smooth with respect to the discovered graph.

The contributions in this paper are threefold. First, we model the graph learning problem as an edge selection problem, where we parameterize the graph through a *sparse edge sampling vector*. In particular, the proposed model provides an elegant handle to control the number of edges, thus the graph sparsity. Second, for the case

when the true smooth graph signals are given, the graph learning problem can be solved optimally, and the solution is based on simple *rank ordering*. Finally, given the noisy graph signals, i.e., for the joint sparse graph learning and denoising problem, we provide a one-step solution based on convex optimization as well as an algorithm based on alternating minimization.

The problem of learning the graph Laplacian or the weighted adjacency matrix from smooth graph signals has been considered before [4,5]. Learning sparse graphs from the true graph signals, which is the problem we consider in Section 3, has been studied in [5]. There the graph learning problem is posed as a constrained optimization problem with the constraint set being the set of valid adjacency matrices, and the optimization problem is solved using an iterative primal-dual algorithm. In contrast, our modelling greatly simplifies the solution to simple *rank ordering*. Such a modelling is inspired from [6], where the problem to design edge weights that maximize the algebraic connectivity of the graph has been addressed. In [4], the joint graph learning and denoising problem has been addressed, i.e., the problem that we study in Section 4. An alternating minimization algorithm is proposed, alternating between graph learning and denoising, where the graph learning optimization problem involves a search over the space of all valid graph Laplacians. On the contrary, we show that this problem can be solved in one-step and it boils down to the design of a sparse edge sampling function.

Graph topology identification is also investigated in [7] under the assumption that the eigenvectors of the graph Laplacian are known, which is a much stronger assumption. Although the eigenvectors can be computed from graph data (or the sample covariance matrix) when it is stationary with respect to the graph [8, 9], the graph signals need not always be vertex stationary. In any case, the estimated eigenvectors are not error free due to limited data records. In most of the existing approaches [4,5,7], graph sparsification is (or can be) achieved by penalizing the ℓ_1-norm of the graph Laplacian matrix, adjacency matrix or the shift operator, however, there is no explicit handle to control the number of edges, unlike the proposed approach. In a related line of research, [10, 11] investigate computing sparse graphs that approximate a given graph spectrally, which means that their Laplacian matrices have similar quadratic forms.

2. PROBLEM SETUP

Consider a dataset with N real valued elements, which are defined on the vertices of an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where the vertex set $\mathcal{V} = \{v_1, \cdots, v_N\}$ denotes the set of nodes, and the edge set $\mathcal{E}$ reveals the connection between the nodes. We refer to such datasets as *graph signals*. We assume that the length of the graph signals (thus the number of nodes), i.e., N is known. However, the edge set is not known. Therefore, we assume a *complete graph* $\mathcal{G}(\mathcal{V}, \mathcal{E})$

as a candidate graph in which each node is connected to every other node with the number of edges $|\mathcal{E}| = M = N(N-1)/2$, and aim to determine a subgraph of $\mathcal{G}$ by choosing a subset of edges, $\mathcal{E}_s$, from the edge set $\mathcal{E}$ of this candidate graph.

Any undirected graph topology is basically determined by its graph Laplacian matrix, which essentially reveals the connectivity of the graph. Let us denote the graph Laplacian matrix (i.e., a symmetric matrix) of the complete graph by $\boldsymbol{L} \in \mathbb{S}^N$, where $[\boldsymbol{L}]_{i,j}$ is nonzero only if $i = j$ or $(i,j) \in \mathcal{E}$. The symmetric matrix $\boldsymbol{L}$ can be expressed in terms of the so-called *incidence matrix*, $\boldsymbol{A} = [\boldsymbol{a}_1, \cdots, \boldsymbol{a}_M] \in \mathbb{R}^{N \times M}$ as

$$\boldsymbol{L} = \boldsymbol{A}\boldsymbol{A}^T = \sum_{m=1}^{M} \boldsymbol{a}_m \boldsymbol{a}_m^T,$$

where the m-th column of $\boldsymbol{A}$, i.e., $\boldsymbol{a}_m$ denotes a length-N edge vector with entries $[\boldsymbol{a}_m]_i = 1$, $[\boldsymbol{a}_m]_j = -1$, and it has zeros elsewhere, for an edge m connecting nodes i with j (more generally, $\boldsymbol{a}_m$ is determined only up to a sign).

Let us now denote the subgraph $\mathcal{G}_s(\mathcal{V}, \mathcal{E}_s)$ with the edge set $\mathcal{E}_s \subset \mathcal{E}$ such that $|\mathcal{E}_s| = K \ll M$. We will refer to such a subgraph with K edges as a K-sparse graph. We connect such a K-sparse graph $\mathcal{G}_s$ to $\boldsymbol{L}$ through a *sparse edge selection* vector $\boldsymbol{w} = [w_1, w_2, \cdots, w_M]^T \in \{0,1\}^M$, where $w_m = 1$ if an edge belongs to the edge subset $\mathcal{E}_s$, and $w_m = 0$ otherwise. In terms of $\boldsymbol{w}$, $|\mathcal{E}_s| = K$ means $\|\boldsymbol{w}\|_0 = K$. (The notation $\|\boldsymbol{w}\|_0$ counts the number of non-zero entries in $\boldsymbol{w}$.) Finally, we can write the Laplacian matrix of the K-sparse graph, $\boldsymbol{L}_s$, as a function of $\boldsymbol{w}$ as

$$\boldsymbol{L}_s(\boldsymbol{w}) = \sum_{m=1}^{M} w_m \boldsymbol{a}_m \boldsymbol{a}_m^T. \tag{1}$$

In what follows, we will optimally design the edge sampling function $\boldsymbol{w}$ to recover the graph that sufficiently explains the data.

3. LEARNING FROM NOISELESS GRAPH SIGNALS

Let $\boldsymbol{x} = [x_1, x_2, \cdots, x_N]^T \in \mathbb{R}^N$ be a graph signal defined on the vertices $\mathcal{V}$ of a graph. The smoothness and the spectral content of the signal both depend on the underlying graph topology. The Laplacian quadratic form given by $\boldsymbol{x}^T \boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{x}$ quantifies how smooth the graph signal $\boldsymbol{x}$ is with respect to the underlying graph [1]. In particular, the signal $\boldsymbol{x}$ is smoothest with respect to the graph with K edges for low values of $\boldsymbol{x}^T \boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{x}$.

3.1. Problem statement: noiseless setting

Suppose we are given L graph signals denoted by the vectors $\{\boldsymbol{x}_k\}_{k=1}^L$, and they are collected in an $N \times L$ matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \cdots, \boldsymbol{x}_L]$. We are interested in recovering the graph Laplacian (in other words, the graph topology) under the prior information that the graph signals are smooth with respect to a K-sparse graph. More formally, we state the following.

Problem 1. *Given the graph signals $\{\boldsymbol{x}_k\}_{k=1}^L$, determine a graph with K edges such that the graph signals have smooth variations on the resulting graph.*

Mathematically, the above problem can be cast as the following optimization problem:

$$\underset{\boldsymbol{w} \in \mathcal{W}}{\arg\min} \quad \frac{1}{L}\sum_{k=1}^{L} \boldsymbol{x}_k^T \boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{x}_k = \frac{1}{L}\text{tr}\{\boldsymbol{X}^T \boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{X}\}, \tag{2}$$

where $\mathcal{W} = \{\boldsymbol{w} \in \{0,1\}^M \,|\, \|\boldsymbol{w}\|_0 = K\}$ is the constraint set that restricts the number of edges.

3.2. Solver

Problem (2) is a cardinality constrained Boolean optimization problem, hence nonconvex. By recalling that $\boldsymbol{L}_s(\boldsymbol{w}) = \sum_{m=1}^{M} w_m \boldsymbol{a}_m \boldsymbol{a}_m^T$, we can express the cost function in (2) as a linear function in $\boldsymbol{w}$, i.e., we have

$$\frac{1}{L}\text{tr}\left\{\boldsymbol{X}^T \boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{X}\right\} = \sum_{m=1}^{M} w_m \text{tr}\left\{\boldsymbol{X}^T(\boldsymbol{a}_m \boldsymbol{a}_m^T)\boldsymbol{X}\right\}. \tag{3}$$

Introducing the length-M vector $\boldsymbol{c} = [c_1, c_2, \ldots, c_M]^T$ with $c_m = \text{tr}\left\{\boldsymbol{X}^T(\boldsymbol{a}_m \boldsymbol{a}_m^T)\boldsymbol{X}\right\}$, we can write (2) as

$$\underset{\boldsymbol{w} \in \{0,1\}^M}{\arg\min} \quad \boldsymbol{c}^T \boldsymbol{w} \quad \text{s.to} \quad \|\boldsymbol{w}\|_0 = K. \tag{4}$$

The above Boolean linear programming problem admits an explicit solution and computing the optimal solution is straightforward. It is solved by sorting the entries of $\boldsymbol{c}$ in an ascending ordering. More specifically, the solution $\boldsymbol{w}$ will have entries equal to 1 at indices corresponding to the K smallest entries of $\boldsymbol{c}$, and others are set to zero (ties may be broken arbitrarily). Computationally, the sorting algorithm costs $\mathcal{O}(K \log K)$, and with a parallel implementation (e.g., on different processors), the computational complexity will be as low as $\mathcal{O}(K)$ [12, 13]. We give another interpretation of this result through the following remark.

Remark 1. *Let us suppose the graph signal is stochastic with covariance matrix $\boldsymbol{R}_{\boldsymbol{x}} = \mathbb{E}\{\boldsymbol{x}\boldsymbol{x}^T\} \in \mathbb{R}^{N \times N}$. Then, the solution to (4) would select K edges between those nodes having the highest cross-correlation, i.e., it will add an edge between the ith and the jth node if the variables x_i and x_j are strongly correlated. To see this, we express the cost function in (2) as*

$$L^{-1}\text{tr}\{\boldsymbol{X}^T \boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{X}\} = \text{tr}\left\{\boldsymbol{L}_s(\boldsymbol{w})\widehat{\boldsymbol{R}}_{\boldsymbol{x}}\right\}$$
$$= \sum_{m=1}^{M} w_m(\boldsymbol{a}_m^T \widehat{\boldsymbol{R}}_{\boldsymbol{x}} \boldsymbol{a}_m)$$

where $\widehat{\boldsymbol{R}}_{\boldsymbol{x}} = \frac{1}{L}\boldsymbol{X}\boldsymbol{X}^T \in \mathbb{R}^{N \times N}$ is the sample data covariance matrix. Recalling the definition of $\boldsymbol{a}_m$, it is easy to see that the term $\boldsymbol{a}_m^T \widehat{\boldsymbol{R}}_{\boldsymbol{x}} \boldsymbol{a}_m = [\widehat{\boldsymbol{R}}_{\boldsymbol{x}}]_{i,i} + [\widehat{\boldsymbol{R}}_{\boldsymbol{x}}]_{j,j} - 2[\widehat{\boldsymbol{R}}_{\boldsymbol{x}}]_{i,j}$ is small if the ith and jth nodes are highly correlated and we have sufficient samples to compute the sample covariance matrix.

By modelling the graph topology through an edge selection vector, the graph learning problem can be solved optimally using a simple and elegant solution with a controlled sparsity level, whereas optimizing directly the graph Laplacian [4] or the adjacency matrix [5] leads to a more complicated suboptimal solution with no explicit handle to control the graph sparsity.

4. LEARNING FROM NOISY GRAPH SIGNALS

In many cases, we might not have access to the true graph signals. Suppose we observe a noisy version of the graph signal, $\boldsymbol{x}_k$, as

$$\boldsymbol{y}_k = \boldsymbol{x}_k + \boldsymbol{n}_k \in \mathbb{R}^N, \tag{5}$$

and we are given L such observations for $k = 1, 2, \ldots, L$, where we assume that $\boldsymbol{n}_k$ is zero-mean white Gaussian noise of variance σ^2.

To recover $\boldsymbol{x}_k$ based on the smoothness assumption, typically a least-squares problem is solved with a *Tikhonov regularization*, $\boldsymbol{x}_k^T \boldsymbol{L} \boldsymbol{x}_k$, to enforce the prior information that the noiseless graph signal $\boldsymbol{x}_k$ is smooth with respect to the underlying graph. More specifically, the following optimization problem (assuming, for a moment that the graph, i.e., $\boldsymbol{w}$ is known) is solved [1]:

$$\underset{\{\boldsymbol{x}_k\}_{k=1}^L}{\arg\min} \quad \frac{1}{L}\sum_{k=1}^L \left(\|\boldsymbol{y}_k - \boldsymbol{x}_k\|_2^2 + \gamma \boldsymbol{x}_k^T \boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{x}_k \right), \quad (6)$$

where the regularization parameter $\gamma > 0$ controls the amount of smoothness. This graph denoising problem has an explicit solution given by

$$\widehat{\boldsymbol{x}}_k = [\boldsymbol{I} + \gamma\boldsymbol{L}_s(\boldsymbol{w})]^{-1}\boldsymbol{y}_k, \; k = 1, \cdots, L.$$

4.1. Problem statement: noisy setting

Having given the above denoising inference problem at hand, we will now formally state the problem of interest.

Problem 2. *Given the observations $\{\boldsymbol{y}_k\}_{k=1}^L$ that are related to the unknown graph signal $\boldsymbol{x}_k$ as in (5), determine the K-sparse graph such that the estimate $\widehat{\boldsymbol{x}}_k$ has the lowest possible estimation error, and it is smooth with respect to the recovered graph.*

Sparse graph learning for the denoising inference problem can be mathematically formulated as follows:

$$\underset{\{\boldsymbol{x}_k\}_{k=1}^L, \boldsymbol{w}\in\mathcal{W}}{\arg\min} \frac{1}{L}\sum_{k=1}^L (\|\boldsymbol{y}_k - \boldsymbol{x}_k\|_2^2 + \gamma\,\boldsymbol{x}_k^T\boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{x}_k) \quad (7)$$

whose solution is denoted as $(\{\widehat{\boldsymbol{x}}_k\}_{k=1}^L, \widehat{\boldsymbol{w}})$. This formulation is different from [4], as [4] solves an optimization problem over the space of all possible graph Laplacians (instead of parameterizing the graph with $\boldsymbol{w} \in \mathcal{W}$) without sparsifying the graph. It can nevertheless be done through an extra ℓ_1-norm penalty term.

The above problem (7) is noncovex due to the Boolean and cardinality constraints on $\boldsymbol{w}$ and the coupling between the optimization variables in the second term of (7). We provide two methods to solve it. The first one is a straightforward approach based on alternating descent, while the second one is based on convex relaxation.

4.2. Alternating minimization

The optimization problem (7) can be solved using alternating minimization with respect to $\{\boldsymbol{x}_k\}_{k=1}^L$ and $\boldsymbol{w}$. That is, given $\boldsymbol{w}$, the problem in (7) reduces to a linear system in the unknown $\boldsymbol{X}$, which admits a closed form solution; while given $\{\boldsymbol{x}_k\}_{k=1}^L$, it reduces to a Boolean linear programming problem, which admits an analytical solution with respect to $\boldsymbol{w}$ based on rank ordering. These observations suggest an iterative alternating minimization algorithm yielding successive estimates of $\{\boldsymbol{x}_k\}_{k=1}^L$ with fixed $\boldsymbol{w}$, and alternately of $\boldsymbol{w}$ with fixed $\{\boldsymbol{x}_k\}_{k=1}^L$. Specifically, with the iterate of $\boldsymbol{w}$ given per iteration $i \geq 0$, i.e., $\boldsymbol{w}[i]$, we solve for $\boldsymbol{X}[i]$ using a matrix inversion as

$$\boldsymbol{X}[i] = \boldsymbol{X}_{\min}(\boldsymbol{w}[i])$$

with

$$\begin{aligned}\boldsymbol{X}_{\min}(\boldsymbol{w}) &= \underset{\boldsymbol{X}}{\arg\min}\, \|\boldsymbol{Y}-\boldsymbol{X}\|_F^2 + \gamma\,\mathrm{tr}\{\boldsymbol{X}^T\boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{X}\}\\ &= [\boldsymbol{I} + \gamma\boldsymbol{L}_s(\boldsymbol{w})]^{-1}\boldsymbol{Y},\end{aligned} \quad (8)$$

where $\boldsymbol{Y} = [\boldsymbol{y}_1, \boldsymbol{y}_2, \cdots, \boldsymbol{y}_L]$ is the data matrix of size $N \times L$.

Once $\boldsymbol{X}[i]$ is available, $\boldsymbol{w}[i+1]$ can be obtained by solving the Boolean linear program [cf. (4)]

$$\boldsymbol{w}[i+1] = \underset{\boldsymbol{w}\in\{0,1\}^M}{\arg\min} \quad \sum_{m=1}^M w_m c_m[i+1] \quad \text{s.to} \quad \|\boldsymbol{w}\|_0 = K,$$

where $c_m[i+1] = \mathrm{tr}\left\{\boldsymbol{X}^T[i+1](\boldsymbol{a}_m\boldsymbol{a}_m^T)\boldsymbol{X}[i+1]\right\}$. In spite of the Boolean and cardinality constraints in the above problem, there exists a simple analytical solution for $\boldsymbol{w}[i+1]$ based on sorting $\{c_m[i+1]\}_{m=1}^M$, i.e., the solution $\boldsymbol{w}[i+1]$ will have entries equal to 1 at indices corresponding to the K smallest entries in $\{c_m[i+1]\}_{m=1}^M$ and zeros otherwise. The iterations are initialized at $i = 0$ by randomly generating $\boldsymbol{w}[i+1]$ from a uniform distribution over $\mathcal{W}$. The above alternating minimization method is computationally very attractive, and consists of two simple known solutions per iteration. However, the algorithm converges only to a stationary point of (7), and it suffers from the choice of the initial estimate.

The algorithm proposed in [4] is also along the lines of alternating minimization, except that the graph learning step involves a complicated optimization over the space of all possible valid Laplacian matrices.

4.3. Convex relaxation

To avoid the issues related to the initialization of the alternating minimization algorithm, in what follows we propose a one-step solution based on convex relaxation. We can rewrite the formulation in (7) alternatively as

$$\widehat{\boldsymbol{w}} = \underset{\boldsymbol{w}\in\mathcal{W}}{\arg\min} \quad r(\boldsymbol{w}); \quad \widehat{\boldsymbol{X}} = \boldsymbol{X}_{\min}(\widehat{\boldsymbol{w}}) \quad (9)$$

with

$$r(\boldsymbol{w}) = \|\boldsymbol{Y} - \boldsymbol{X}_{\min}(\boldsymbol{w})\|_F^2 + \gamma\,\mathrm{tr}\{\boldsymbol{X}_{\min}^T(\boldsymbol{w})\boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{X}_{\min}(\boldsymbol{w})\}$$

and

$$[\boldsymbol{I} + \gamma\boldsymbol{L}_s(\boldsymbol{w})]\boldsymbol{X}_{\min}(\boldsymbol{w}) = \boldsymbol{Y}. \quad (10)$$

The computational complexity of solving the linear system of equations (10) decreases as the sparsity in $\boldsymbol{w}$ increases. Furthermore, the estimates $\{\widehat{\boldsymbol{x}}_k\}_{k=1}^L$ and $\widehat{\boldsymbol{w}}$ in (9) are still the same as in (7).

Plugging the solution to (10) in $r(\boldsymbol{w})$ and after some straightforward matrix algebra, we can express the regularized residual squared, $r(\boldsymbol{w})$, as
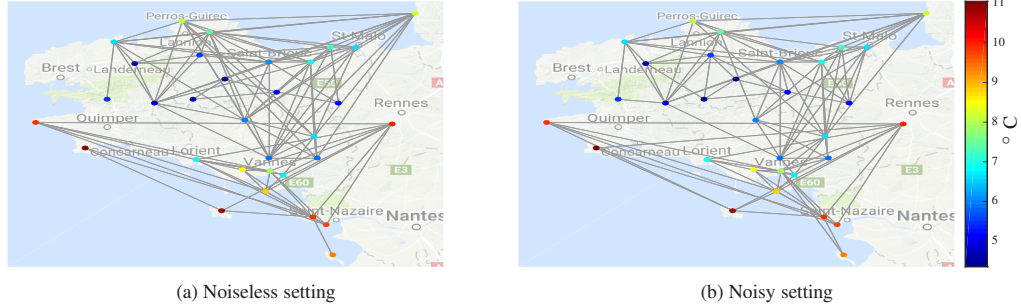
$$\begin{aligned}r(\boldsymbol{w}) = \mathrm{tr}\left\{\boldsymbol{Y}^T[\boldsymbol{I} + \gamma\boldsymbol{L}_s(\boldsymbol{w})]^{-1}\boldsymbol{Y}\right\}\\ + \gamma\mathrm{tr}\left\{\boldsymbol{Y}^T\boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{Y}\right\} - \|\boldsymbol{Y}\|_F^2.\end{aligned} \quad (11)$$

Relaxing the cardinality constraint $\|\boldsymbol{w}\|_0 = K$ with $\boldsymbol{1}^T\boldsymbol{w} = K$ and the Boolean constraints $\{0,1\}^M$ with linear inequality constraints related to the box constraint $[0,1]^M$, the optimization problem (9) will be convex on $\boldsymbol{w} \in [0,1]^M$. To see this, we introduce a variable

$$\boldsymbol{Z} = \boldsymbol{Y}^T[\boldsymbol{I} + \gamma\boldsymbol{L}_s(\boldsymbol{w})]^{-1}\boldsymbol{Y} + \gamma\boldsymbol{Y}^T\boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{Y} \in \mathbb{R}^{L\times L}$$

and obtain a semidefinite program:

$$\begin{aligned}&\underset{\boldsymbol{Z},\boldsymbol{w}}{\arg\min} \quad \mathrm{tr}\{\boldsymbol{Z}\}\\ &\text{s.to} \quad \begin{bmatrix}\boldsymbol{Z} - \gamma\boldsymbol{Y}^T\boldsymbol{L}_s(\boldsymbol{w})\boldsymbol{Y} & \boldsymbol{Y}^T\\ \boldsymbol{Y} & \boldsymbol{I}+\gamma\boldsymbol{L}_s(\boldsymbol{w})\end{bmatrix} \succeq \boldsymbol{0}_{L+N}, \quad (12)\\ &\quad\quad \boldsymbol{1}^T\boldsymbol{w} = K, \; 0 \leq w_m \leq 1, m = 1,2,\ldots,M,\end{aligned}$$

(a) Noiseless setting

(b) Noisy setting

Fig. 1: *Sparse graph learning*: The colored dots indicate the temperature values. (a) Noiseless case. Graph with $K = 110$ edges recovered by solving (2). (b) Noisy case: Convex relaxation is used to recover a graph with $K = 110$ edges using (12).
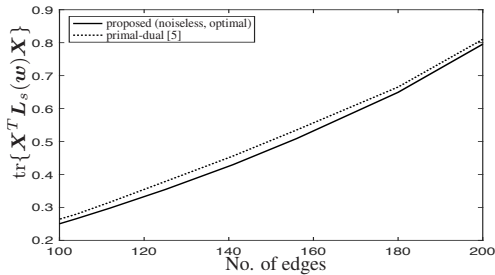
(a) Noiseless setting: smoothness for different graphs
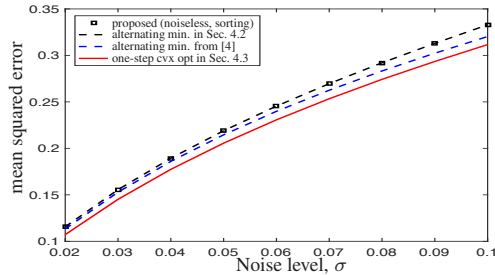
(b) Denoising error for different noise levels

Fig. 2: Performance evaluation.

with variables $\boldsymbol{w}$ and $\boldsymbol{Z}$, and recall that $\boldsymbol{L}_s(\boldsymbol{w}) = \sum_{m=1}^{M} w_m \boldsymbol{a}_m \boldsymbol{a}_m^T$. A standard off-the-shelf solver can be used for solving the semidefinite program in (12). For large-scale problems, computationally cheaper first-order (and online) methods for solving (12) can be derived as the size of the linear matrix inequality in (12) depends on the size of the training data and the number of nodes.

5. NUMERICAL RESULTS

We use temperature measurements collected across 32 weather stations in the French region of Brittany and the aim is to learn the graph that explains the observed data; see Fig. 1. There are 744 observations per weather station available, out of which we use $L = 50$ snapshots as the training set and the remaining ones as the evaluation set. One such observation (i.e., a graph signal) on a graph with $N = 32$ nodes is shown in Fig. 1, where the colored dots indicate different temperature readings. The convex optimization problems are solved using the CVX toolbox, which internally calls

SDPT3 [14]. The candidate graph with $N = 32$ will have $M = 496$ edges, from which we aim to learn a subgraph with $K = 110$ edges.

To begin with, we consider the noiseless case, where the true graph signal is assumed to be known, and graph learning in this case amounts to solving a sorting problem. As shown in Fig. 1a, we can see that in the learnt graph with $K = 110$ edges, edges are present between nodes that share similar values. Although the proposed approach doesn't always (e.g., for low values of K) ensure a well-connected graph, it clusters entities (or correlated nodes) with similar values. Fig. 2a shows that the cost (i.e., smoothness) of the proposed closed-form sorting solution, which is optimal, is lower than the existing iterative solution [5].

Next, we consider the noisy setting with the same training data as before, where we perform joint graph learning and denoising. In Fig. 1b, we show the learnt graph with $K = 110$ edges based on the convex relaxation approach explained in Sec. 4.3. In Fig. 2b, we evaluate the denoising performance based on the learnt graph using the evaluation set. In particular, we show the mean squared error for different values of the noise level, where the mean squared error is computed from 1000 independent Monte Carlo experiments. The one-step solution based on convex optimization (cf. Sec. 4.3) leads to a lower error as compared to the alternating minimization approaches, which in general converge only to a stationary point. This also holds for our method developed in Sec. 4.2, however, we stress the fact that the proposed alternating minimization (cf. Sec. 4.2) is computationally much less expensive (involving two simple known solutions per iteration) as compared to the iterative solution in [4]. The graph learnt under the noiseless setting does not perform well for denoising. Nevertheless, due its simple solution, it can be used to generate a base graph, which can be further refined for specific graph inference problems.

6. CONCLUSIONS

We have studied the problem of learning a sparse graph that adequately explains the data under a smoothness prior. We model the graph learning problem as the design of a sparse edge sampling function. In other words, we express the graph Laplacian in terms of an edge selection vector. We have considered both the noiseless and noisy setting. In the noiseless setting, designing the edge selection vector is elegant, and it boils down to a simple low-complexity sorting problem. However, in the presence of noise, we propose a computationally cheap alternating minimization algorithm as well as a one-step convex relaxation based solution.

7. 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 Process. Mag.*, vol. 30, no. 3, pp. 83–98, 2013.

[2] A. Sandryhaila and J. M. Moura, "Big data analysis with signal processing on graphs: Representation and processing of massive data sets with irregular structure," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 80–90, 2014.

[3] K. Slavakis, G. Giannakis, and G. Mateos, "Modeling and optimization for big data analytics:(statistical) learning tools for our era of data deluge," *IEEE Signal Process. Mag.*, vol. 31, no. 5, pp. 18–31, 2014.

[4] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *arXiv preprint arXiv:1406.7842*, 2014.

[5] V. Kalofolias, "How to learn a graph from smooth signals," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, 2016, pp. 920–929.

[6] A. Ghosh and S. Boyd, "Growing well-connected graphs," in *Proceedings of the 45th IEEE Conference on Decision and Control*, 2006, pp. 6605–6611.

[7] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology identification from spectral templates," *arXiv preprint arXiv:1604.02610*, 2016.

[8] A. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Stationary graph processes and spectral estimation," *arXiv preprint arXiv:1603.04667*, 2016.

[9] S. P. Chepuri and G. Leus, "Subsampling for graph power spectrum estimation," in *IEEE Sensor Array and Multichannel Signal Processing Workshop (SAM)*, July 2016, pp. 1–5.

[10] D. A. Spielman and S.-H. Teng, "Spectral sparsification of graphs," *SIAM Journal on Computing*, vol. 40, no. 4, pp. 981–1025, 2011.

[11] J. Batson, D. A. Spielman, and N. Srivastava, "Twice-ramanujan sparsifiers," *SIAM Journal on Computing*, vol. 41, no. 6, pp. 1704–1721, 2012.

[12] R. S. Blum and B. M. Sadler, "Energy efficient signal detection in sensor networks using ordered transmissions," *IEEE Trans. Signal Process.*, vol. 56, no. 7, pp. 3229–3235, 2008.

[13] S. P. Chepuri and G. Leus, "Sparse sensing for distributed detection," *IEEE Trans. Signal Process.*, vol. 64, no. 6, pp. 1446–1460, 2015.

[14] M. Grant and S. Boyd, "CVX: Matlab software for disciplined convex programming, version 2.0 beta," http://cvxr.com/cvx, Sep. 2012.