ONLINE GRAPH LEARNING FROM SEQUENTIAL DATA

Stefan Vlaski, Hermina P. Maretić, Roula Nassif, Pascal Frossard and Ali H. Sayed
Institute of Electrical Engineering, EPFL, Lausanne, Switzerland

ABSTRACT

Graphs provide a powerful framework to represent high-dimensional but structured data, and to make inferences about relationships between subsets of the data. In this work we consider graph signals that evolve dynamically according to a heat diffusion process and are subject to persistent perturbations. We develop an online algorithm that is able to learn the underlying graph structure from observations of the signal evolution. The algorithm is adaptive in nature and in particular able to respond to changes in the graph structure and the perturbation statistics.

Index Terms— Graph learning, Online learning, Laplacian matrix, Adaptive algorithm.

1. INTRODUCTION

In data science applications, effective interpretation and processing of high-dimensional data is generally contingent on an understanding of the relationships that may exist between subsets of the data. This is particularly relevant for large-scale data sets. One useful way to capture interrelations among different parts of a data set is by means of a graph representation or model [1]. While data arising from some applications naturally lead to or suggest suitable graph representations for information flow, such as graphs representing networks or power grids, there are many instances where the underlying graph structure is not readily available and needs to be inferred from observations. Furthermore, even when the topology of the graph is known, the same may not hold for the weights on the edges of the graph, which describe the *strength* of the relationship. In the example of a social network for example, it may be less important to know whether two people are connected, than to know how much influence one person has on the other.

In this work, we consider signals that evolve according to a heat diffusion process [2]. This process is related to a spatially sampled approximation of the second-order heat differential equation. The model is not limited to heat diffusion but can be applied to modeling other processes such as the evolution of interest over social networks [3] and the movement of people in cities [4]. We shall show that the problem of recovering the graph Laplacian, which parametrizes the heat diffusion process, from the time evolution of the observed signal, can be formulated as a strongly-convex and quadratic optimization problem. This in turn means that its minimizer can be sought efficiently by a variety of algorithms. We propose a (projected) stochastic gradient algorithm, which amounts to a Least-Mean-Squares (LMS)-type recursion and is adaptive in nature.

2. RELATED WORKS

The earliest works related to graph learning are based on *sparse* estimation of precision matrices, i.e., inverse covariance matrices [5, 6]. The work in [7] introduced structural constraints to ensure that the learned (regularized Laplacian) matrix describes a valid graph. A string of subsequent works [8, 9, 10] leverage the concept of a "smooth signal over a graph". The drawback of these approaches is that the smoothness assumption may not be satisfied in some important applications, particularly if the graph signal is dynamic or perturbed by events on the graph.

The interpretation of graph-shifts as a generalization of the traditional shift operation in digital signal processing has motivated a number of generalizations of DSP concepts to the graph domain. Autoregressive graph filters in terms of polynomials of the adjacency matrix are used in [11] to model the signal evolution over the graph and infer the adjacency matrix. The heat diffusion model is considered in [4], where an algorithm is proposed to leverage a collection of *independent* samples which are modeled as the superposition of a small number of perturbations that diffuse over the graph.

Both of these recent works allow for dynamic signals that evolve according to some graph topology that is subsequently learned. This is achieved by collecting all available samples and solving an optimization problem based on a batch of data. As such, even though the model allows for dynamic signals, the algorithms themselves are not dynamic; the underlying assumption is that the model parameters are fixed. In contrast, in this work, we develop a truly adaptive solution that responds to streaming data and has the potential to track drifts in both the graph and data statistics under the heat diffusion model. Dynamic algorithms for the estimation of edge propabilities in social interations are developed in [12, 13] and for autoregressive graph processes in [14].

3. FRAMEWORK

3.1. Graph Model

We consider weighted, undirected graphs without self-loops. Every pair of vertices i and j is assigned a weight $a_{ij} = a_{ji}$, which quantifies their relative influence, in a manner made precise in the signal model further below. We collect these weights into an adjacency matrix $A = [a_{ij}]$ that satisfies the following properties:

$$\text{Symmetry: } A = A^{\mathsf{T}} \tag{1}$$

$$\text{Non-negativity: } a_{ij} \geq 0, \ \forall\, i, j \tag{2}$$

$$\text{No self-loops: } a_{ii} = 0, \ \forall\, i \tag{3}$$

A common and useful matrix to describe and study graphs is the Laplacian matrix, defined as:

$$L \triangleq \text{diag}\left(A\mathbb{1}\right) - A \tag{4}$$

The work of A. H. Sayed was supported in part by NSF grants CCF-1524250 and ECCS-1407712. Emails: {stefan.vlaski, hermina.petricmaretic, roula.nassif, pascal.frossard, ali.sayed}@epfl.ch

Under conditions (1)–(3) on the adjacency matrix, the graph Laplacian L satisfies the following properties [15]:

$$\text{Symmetry: } L = L^\mathsf{T} \tag{5}$$

$$\text{Non-positive off-diagonal elements: } \ell_{ij} \leq 0,\ \forall\, i \neq j \tag{6}$$

$$\text{Positive definite: } L \succeq 0 \tag{7}$$

$$\text{Nullspace: } L\frac{1}{\sqrt{N}}\mathbb{1} = 0 \tag{8}$$

3.2. Signal Model

We shall assume that we observe discrete samples of a continuous time graph process $s(t) \in \mathbb{R}^N$, which evolves according to the differential equation [2]:

$$s'(t) = -L^\star s(t) + p(t) \tag{9}$$

where $L^\star \in \mathbb{R}^{N \times N}$ denotes the Laplacian matrix of the underlying graph linking the entries of $s(t)$, and $p(t) \in \mathbb{R}^N$ describes a process that drives the signal dynamics. The variable $p(t)$ can either be viewed as an outside force, which influences the evolution of the signal, or some internal events that subsequently diffuse over the graph. The solution to the differential equation (9) has the form:

$$s(t) = e^{-tL^\star} s(0) + \int_0^t e^{-(t-u)L^\star} p(u) du \tag{10}$$

We have access to the evolution of the graph signal beginning at some time t_0 and subsequently at times $t_i = t_0 + iT, i > 0$, where $i \in \mathbb{N}$ denotes the i-th sample and $T \in \mathbb{R}_{>0}$ denotes the sampling period. We observe a recursive relationship between adjacent samples, that is critical for this work, namely the fact that:

$$s(t_i) = e^{-TL^\star} s(t_{i-1}) + \int_{t_i - T}^{t_i} e^{-(t_i-u)L^\star} p(u) du \tag{11}$$

Note that the relationship between $s(t_i)$ and $s(t_{i-1})$ only depends on $L^\star$ and on the perturbations $p(t)$ between t_i and t_{i-1}. We move into the discrete domain by letting $s_i \triangleq s(t_0 + iT)$ and $p_i \triangleq \int_{t_i-T}^{t_i} e^{-(t_i-u)L^\star} p(u) du$ so that (11) becomes:

$$s_i = e^{-TL^\star} s_{i-1} + p_i \tag{12}$$

Since we are generally not provided with the perturbations that drive the system, we shall model the driving term p_i as a stochastic random variable, so that:

$$\boldsymbol{s}_i = e^{-TL^\star}\, \boldsymbol{s}_{i-1} + \boldsymbol{p}_i \tag{13}$$

where we are using boldface notation to refer to random variables.

The objective of this work is to develop a solution that allows for the estimation of $L^\star$ from streaming realizations $\boldsymbol{s}_i$. These types of algorithms generally operate by evaluating the prediction error of the current estimate on the incoming observation and adjusting the estimate based on this error. Under the non-linear model (13), every such iteration requires the evaluation of a matrix exponential and is computationally expensive. This is particularly critical in scenarios where the graph size is large.

3.3. An Equivalent Linear Model

On the face of it, it is straightforward to define

$$W^\star \triangleq e^{-TL^\star} \tag{14}$$

so that the relation becomes

$$\boldsymbol{s}_i = W^\star\, \boldsymbol{s}_{i-1} + \boldsymbol{p}_i \tag{15}$$

However, it is important to remember that $L^\star$ is a Laplacian matrix and hence required to satisfy properties (5)–(8). It turns out that an equivalent set of properties can be imposed on $W^\star$ to ensure that $L^\star = \frac{-1}{T} \ln(W^\star)$ describes a valid Laplacian matrix and hence a valid graph. To begin with, we introduce the eigendecomposition of the Laplacian matrix:

$$L^\star = V \Lambda_L V^\mathsf{T} \tag{16}$$

Expanding the matrix exponential as an infinite sum and recalling that $VV^\mathsf{T} = I$, we obtain:

$$\begin{aligned} W^\star = e^{-TL^\star} &= \sum_{k=0}^{\infty} \frac{(-T)^k}{k!} (L^\star)^k = \sum_{k=0}^{\infty} \frac{(-T)^k}{k!} \left(VL^\star\Lambda^\mathsf{T}\right)^k \\ &= V \left(\sum_{k=0}^{\infty} \frac{(-T)^k}{k!} \Lambda^k\right) V^\mathsf{T} = Ve^{-T\Lambda}V^\mathsf{T} \end{aligned} \tag{17}$$

where $e^{-T\Lambda} = \text{diag}\left\{e^{-T\lambda_k(L^\star)}\right\}$ since Λ is diagonal. This means that the matrix exponential preserves the set of eigenvectors of $L^\star$ and there is a simple relationship between the eigenvalues of $W^\star$ and $L^\star$. This relation also provides a method for calculating the matrix logarithm. Given the eigendecomposition $W^\star = V\Lambda_W V^\mathsf{T}$, the logarithm is given by $\ln(W^\star) = V \ln(\Lambda_W) V^\mathsf{T}$, where $\ln(\Lambda_W) = \text{diag}\{\ln(\lambda_k(W^\star))\}$. This allows us to establish the following conditions on $W^\star$ to ensure that $L^\star$ describes a valid graph.

Lemma 1 (Conditions on $W^\star$)**.** *Let $W \in \mathbb{R}^{N\times N}$ and $L = \frac{-1}{T}\ln(W)$. Then, for sufficiently small sampling periods T, L is a valid Graph Laplacian* if, and only if, *W satisfies the following properties:*

$$\textit{Symmetry: } W = W^\mathsf{T} \tag{18}$$

$$\textit{Non-negative elements: } w_{ij} \geq 0,\ \forall\, i, j \tag{19}$$

$$\textit{Spectral bound: } I \succeq W \succ 0 \tag{20}$$

$$\textit{Stochastic: } W\mathbb{1} = \mathbb{1} \tag{21}$$

Proof. Properties (18), (20) and (21) follow from (17), while (19) follows from the fact the $-L$ is a Metzler matrix [16]. Details omitted due to space limitations. □

3.4. Graph Signal Evolution

Observe that since $\rho(W^\star) = 1$, the recursion described by (15) is not mean-square stable. This means that, while the recursion will converge in the mean as long as $\mathbb{E}\,\boldsymbol{p}_i = 0$, the same does not hold for covariance matrix of $\boldsymbol{s}_i$. It turn out, however, that the centered signal across the graph is mean-square stable as long as the graph is connected. We make this statement precise in the following.

Assumption 1 (Connected graph)**.** *The graph described by A and L is connected. In other words, there is a path of non-zero weights from any vertex to any other vertex in the graph.*

It then follows that the eigenvalue at zero has multiplicity one with unique (normalized) eigenvector $\frac{1}{\sqrt{N}}\mathbb{1}$ [15]. A direct consequence of this property is that the graph Laplacian has a particular eigenstructure $L^\star = V\Lambda_L V^\mathsf{T}$ where:

$$V = \begin{bmatrix} \frac{1}{\sqrt{N}}\mathbb{1} & \overline{V} \end{bmatrix}, \quad \Lambda_L = \begin{bmatrix} 0 & 0 \\ 0 & \overline{\Lambda}_L \end{bmatrix} \tag{22}$$

and critically $\overline{\Lambda}_L$ is *strictly* positive definite:

$$\overline{\Lambda}_L \succ 0 \tag{23}$$

The driving matrix $W^\star$ inherits a similar structure from $L^\star$ via (17). In particular, we have $W^\star = V\Lambda_W V^\mathsf{T}$, where:

$$V = \begin{bmatrix} \frac{1}{\sqrt{N}}\mathbb{1} & \overline{V} \end{bmatrix}, \quad \Lambda_W = \begin{bmatrix} 1 & 0 \\ 0 & \overline{\Lambda}_W \end{bmatrix} \tag{24}$$

and $\overline{\Lambda}_W = e^{-T\overline{\Lambda}_L}$, which due to (23) implies that

$$0 \prec \overline{\Lambda}_W \prec I \tag{25}$$

so that $\rho\left(\overline{\Lambda}_W\right) < 1$. The mean across the graph of the signal at time i is given by $\boldsymbol{s}_{c,i} = \frac{1}{N}\mathbb{1}^\mathsf{T}\boldsymbol{s}_i$. Subtracting this mean yields the centered graph signal $\overline{\boldsymbol{s}}_i$:

$$\overline{\boldsymbol{s}}_i \triangleq \boldsymbol{s}_i - \boldsymbol{s}_{c,i} = \left(I - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}\right)\boldsymbol{s}_i \tag{26}$$

It is important to recognize that the mean contains no information about the graph. This is because for any doubly stochastic W:

$$W\,\boldsymbol{s}_i = W\left(\overline{\boldsymbol{s}}_i + \mathbb{1}\otimes\boldsymbol{s}_{c,i}\right) = W\overline{\boldsymbol{s}}_i + \mathbb{1}\otimes\boldsymbol{s}_{c,i} \tag{27}$$

In other words, the mean is passed through independently of W. For the evolution of the centered signal, we can now write:

$$\overline{\boldsymbol{s}}_i = \overline{W}^\star\overline{\boldsymbol{s}}_{i-1} + \overline{\boldsymbol{p}}_i \tag{28}$$

where we defined:

$$\overline{W}^\star \triangleq W^\star - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}, \quad \overline{\boldsymbol{p}}_i \triangleq \left(I - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}\right)\boldsymbol{p}_i \tag{29}$$

The eigendecomposition of $\overline{W}^\star = V\Lambda_{\overline{W}}V^\mathsf{T}$ is related to the decomposition of $W^\star$ via

$$V = \begin{bmatrix} \frac{1}{\sqrt{N}}\mathbb{1} & \overline{V} \end{bmatrix}, \quad \Lambda_{\overline{W}} = \begin{bmatrix} 0 & 0 \\ 0 & \overline{\Lambda}_W \end{bmatrix} \tag{30}$$

so that the only change is the replacement of the eigenvalue at 1 by 0 and critically now $\rho\left(\overline{W}^\star\right) < 1$. We can examine in detail the evolution of the first and second-order statistics of $\overline{\boldsymbol{s}}_i$.

Assumption 2 (Statistics of the Perturbation Terms)**.** *The statistics of the centered perturbations $\overline{\boldsymbol{p}}_i = \left(I - \frac{1}{N}\mathbb{1}\mathbb{1}^\mathsf{T}\right)\boldsymbol{p}_i$ satisfy the following two conditions for all i:*

$$\mathbb{E}\,\overline{\boldsymbol{p}}_i = 0 \tag{31}$$

$$\mathbb{E}\,\overline{\boldsymbol{p}}_i\overline{\boldsymbol{p}}_i^\mathsf{T} = R_{\overline{\boldsymbol{p}}} < \infty \tag{32}$$

Furthermore, the perturbation $\overline{\boldsymbol{p}}_i$ at time i is independent of $\overline{\boldsymbol{p}}_{i-k}$ for $k > 0$.

Lemma 2 (Signal evolution)**.** *Suppose the network is initially at rest, i.e., $s_0 = 0$ and denote $\mathbb{E}\,\overline{\boldsymbol{p}}_i\overline{\boldsymbol{p}}_i^\mathsf{T} = R_{\overline{\boldsymbol{p}}}$. Then, the first and second-order statistics of the graph process described by* (13) *evolve according to:*

$$\mathbb{E}\,\overline{\boldsymbol{s}}_i = 0 \tag{33}$$

$$\mathbb{E}\,\overline{\boldsymbol{s}}_i\overline{\boldsymbol{s}}_i^\mathsf{T} \triangleq R_{\overline{\boldsymbol{s}}_i} = \sum_{k=0}^{i-1}\left(\overline{W}^\star\right)^{i-k}R_{\overline{\boldsymbol{p}}}\left(\overline{W}^\star\right)^{i-k} \tag{34}$$

Furthermore, the second-order moment converges and we have:

$$\lim_{i\to\infty} R_{\overline{\boldsymbol{s}}_i} \triangleq R_\infty \tag{35}$$

where R_∞ is the solution to the discrete Lyapunov equation:

$$R_{\overline{\boldsymbol{p}}} = R_\infty - \overline{W}^\star R_\infty \overline{W}^\star \tag{36}$$

Proof. Omitted due to space limitations. □

4. GRAPH LEARNING

We now formulate the following optimization problem for learning $\overline{W}^\star$:

$$\overline{W}^\star = \underset{\overline{W}\in\mathcal{C}}{\arg\min}\,\frac{1}{2}\,\mathbb{E}\left\|\overline{\boldsymbol{s}}_i - \overline{W}\overline{\boldsymbol{s}}_{i-1}\right\|^2 \triangleq \underset{\overline{W}\in\mathcal{C}}{\arg\min}\,\mathbb{E}\,J_i\left(\overline{W}\right) \tag{37}$$

where $\mathcal{C}$ is a constraint-set. The cost $J_i(\cdot)$ depends on i because the statistics of $\boldsymbol{s}_{i-1}$ evolve as described in the previous lemma. A natural construction is to choose $\mathcal{C}$ to be the set of matrices that result in a valid Laplacian matrix. It turns out, however, that this is not necessary since $J_i(\overline{W})$ is strongly-convex.

Lemma 3 (Properties of the cost)**.** *The cost specified in* (37) *is Lipschitz continuous and strongly-convex. Specifically, for all $\overline{W} \in \mathbb{R}^{N\times N}$, we have:*

$$J_i(\overline{W}) \geq \frac{\nu_i}{2}\|\overline{W}^\star - \overline{W}\|^2 + \frac{1}{2}\mathrm{Tr}\left(R_{\overline{\boldsymbol{p}}}\right) \tag{38}$$

$$J_i(\overline{W}) \leq \frac{\delta_i}{2}\|\overline{W}^\star - \overline{W}\|^2 + \frac{1}{2}\mathrm{Tr}\left(R_{\overline{\boldsymbol{p}}}\right) \tag{39}$$

where

$$\delta_i = \lambda_{\max}\left(R_{\overline{\boldsymbol{s}}_{i-1}}\right),\; \nu_i = \lambda_{\min}\left(R_{\overline{\boldsymbol{s}}_{i-1}}\right) \tag{40}$$

Moreover, $\overline{W}^\star$ defined in (29) *is the unique minimizer of $J_i(\overline{W})$ for all i.*

Proof. Omitted due to space limitations. □

It follows from this property that the enforcement of properties of $\overline{W}^\star$ is in fact not necessary when designing algorithms for the solution of (37), since any algorithm that converges to a minimizer of (37) will converge to its unique minimizer, $\overline{W}^\star$, which by definition already satisfies all properties that lead to a valid graph Laplacian. Of course, it is reasonable to believe that the addition of constraints and regularization may lead to an increased rate of convergence and/or improved performance in steady-state at the cost of increased computational cost per iteration.

To begin with, we shall pursue the minimizer of (37) in the absence of constraints by means of a stochastic gradient descent algorithm.

Algorithm 1: Laplacian LMS Strategy

$$\overline{\boldsymbol{W}}_i = \overline{\boldsymbol{W}}_{i-1} + \mu\left(\overline{\boldsymbol{s}}_i - \overline{\boldsymbol{W}}_{i-1}\,\overline{\boldsymbol{s}}_{i-1}\right)\overline{\boldsymbol{s}}_{i-1}^\mathsf{T} \tag{41}$$

It is essentially a matrix valued variation of the least-mean squares (LMS) algorithm. To derive approximate expressions for its performance, we shall adopt an assumption on the step-size μ, which is common in the literature [17].

Assumption 3 (Small step-size and independence)**.** *Assume the step-size μ is sufficiently small, so that in the limit, $\|\overline{W}^\star - \overline{\boldsymbol{W}}_i\|^2$ reaches a steady-state distribution and $\overline{W}^\star - \overline{\boldsymbol{W}}_i$ is independent of $\overline{\boldsymbol{s}}_i$.*

Theorem 1 (Performance for small step-sizes)**.** *Under Assumption 3, the mean-square deviation of the estimate from the true minimizer $\overline{W}^\star$ is given by:*

$$\lim_{i\to\infty}\mathbb{E}\left\|\overline{W}^\star - \overline{\boldsymbol{W}}_i\right\|^2 \approx \mu\frac{N\mathrm{Tr}\left(R_{\overline{\boldsymbol{p}}}\right)}{2} \tag{42}$$

Proof. The proof is a straight-forward extension of the arguments given in [17] for classical LMS under the independence assumption. □

Performance of the algorithm can be improved by including projections in the update relation. Recall that W is obtained from $\overline{W}$ via $W = \overline{W} + \frac{1}{N}\mathbb{1}\mathbb{1}^{\mathsf{T}}$. This means that a necessary condition for the properties from Lemma 1 to be satisfied is:

$$\overline{\boldsymbol{W}}_i \in \mathcal{C}_{\text{ele}} \cup \mathcal{C}_{\text{sym}} \cup \mathcal{C}_{\text{null}} \cup \mathcal{C}_{\text{spec}} \tag{43}$$

$$\mathcal{C}_{\text{ele}} \triangleq \left\{ \overline{W} \middle| \overline{w}_{ij} \geq -\frac{1}{N} \right\} \qquad \mathcal{C}_{\text{sym}} \triangleq \left\{ \overline{W} \middle| \overline{W} = \overline{W}^{\mathsf{T}} \right\}$$

$$\mathcal{C}_{\text{null}} \triangleq \left\{ \overline{W} \middle| \overline{W}\mathbb{1} = 0 \right\} \qquad \mathcal{C}_{\text{spec}} \triangleq \left\{ \overline{W} \middle| 0 \preceq \overline{W} \preceq I \right\}$$

Projections onto each of these sets can be evaluated in closed form:

$$\left[\text{Proj}_{\mathcal{C}_{\text{ele}}}\left(\overline{W}\right)\right]_{ij} = \begin{cases} \overline{w}_{ij}, & \text{if } \overline{w}_{ij} \geq -\frac{1}{N} \\ -\frac{1}{N}, & \text{otherwise} \end{cases} \tag{44}$$

$$\text{Proj}_{\mathcal{C}_{\text{sym}}}\left(\overline{W}\right) = \frac{1}{2}\left(\overline{W} + \overline{W}^{\mathsf{T}}\right) \tag{45}$$

$$\text{Proj}_{\mathcal{C}_{\text{null}}}\left(\overline{W}\right) = \overline{W} - \frac{1}{N}\overline{W}\mathbb{1}\mathbb{1}^{\mathsf{T}} \tag{46}$$

$$\text{Proj}_{\mathcal{C}_{\text{spec}}}\left(\overline{W}\right) = V\Lambda_t V^{\mathsf{T}} \tag{47}$$

where the last projection is given in terms of the eigendecomposition of the argument $\overline{W} = V\Lambda V^{\mathsf{T}}$ by thresholding the eigenvalues:

$$[\Lambda_t]_{ii} = \begin{cases} 0, & \text{if } [\Lambda]_{ii} < 0 \\ [\Lambda]_{ii}, & \text{if } 0 \leq [\Lambda]_{ii} \leq 1 \\ 1, & \text{otherwise} \end{cases} \tag{48}$$

We can now interlace these projections with the stochastic gradient update to obtain two algorithms, which explicitly incorporate the structural constraints. Note that the first three projections (44)–(46) are simple in the sense that they require $O(N^2)$ operations where N is the size of the graph, whereas (47) requires a full eigenvalue decomposition. Hence, we can formulate two projected variants of the algorithm. The Type I implementation only enforces simple projections, while Type II enforces all properties.

Algorithm 2: Projected Laplacian LMS Strategy I and II

$$\overline{\boldsymbol{W}}'_i = \overline{\boldsymbol{W}}_{i-1} + \mu\left(\overline{\boldsymbol{s}}_i - \overline{\boldsymbol{W}}_{i-1}\overline{\boldsymbol{s}}_{i-1}\right)\overline{\boldsymbol{s}}_{i-1}^{\mathsf{T}} \tag{49}$$

$$\overline{\boldsymbol{W}}''_i = \text{Proj}_{\mathcal{C}_{\text{sym}}}\left(\text{Proj}_{\mathcal{C}_{\text{null}}}\left(\text{Proj}_{\mathcal{C}_{\text{ele}}}\left(\overline{\boldsymbol{W}}'_i\right)\right)\right) \tag{50}$$

$$\overline{\boldsymbol{W}}_i = \begin{cases} \overline{\boldsymbol{W}}''_i, & \text{for Type I} \\ \text{Proj}_{\mathcal{C}_{\text{spec}}}\left(\overline{\boldsymbol{W}}''_i\right), & \text{for Type II.} \end{cases} \tag{51}$$

Whenever an estimate of the graph Laplacian is required, it is obtained via:

$$\widehat{L} = \frac{-1}{T}\ln\left(\overline{\boldsymbol{W}}_i\right) \tag{52}$$

5. SIMULATION RESULTS

We illustrate the performance of the algorithm in recovering $\overline{W}^{\star}$ as well as the graph structure on a network with $N = 30$ nodes. The perturbation terms are modeled as following a normal distribution with $\boldsymbol{p}_i \sim \mathcal{N}(0, I)$ and the sampling period is $T = 1$. The observations $\boldsymbol{s}_i$ are generated according to (13) and processed according to the algorithms developed in this work. The true graphs is generated using the Barabasi-Albert model [18], upon which random weights between 0.1 and 1.0 are attached to each non-zero edge. After $500,000$ iterations, there is a sudden change in the network topology, to illustrate the ability of the algorithm to adapt. The second graph and its adjacency matrix are depicted in Fig. 1. In the graph representation, small weights are depicted as thin and light lines, while strong weights are dark and thick. Bright colors in the adjacency matrix correspond to large weights.

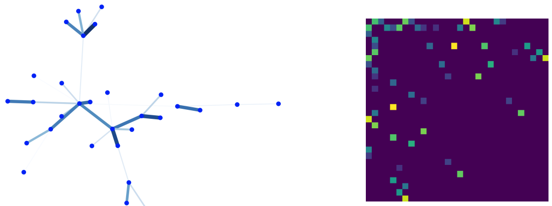

Fig. 1: True graph and adjacency matrix.

The recovered graph and adjacency matrix at the final iteration using Algorithm 2 Type I are depicted in Fig. 2. Color and weight maps are the same as in the representation of the true graph. Key connections along with their weights and the general structure of the graph are accurately recovered. Note that no weights are truly set to zero, resulting in a number of low-weight connections. This is due to the fact that no sparsity prior was imposed on the weight matrix. If desired, they can be removed during post-processing via simple thresholding.

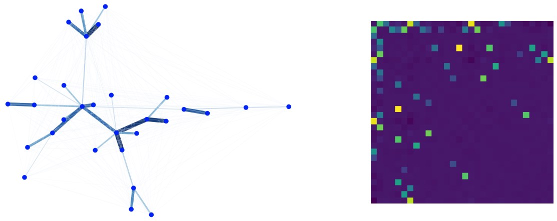

Fig. 2: Recovery using the Projected Laplacian LMS Strategy I.

The mean-square deviation of $\overline{\boldsymbol{W}}_i$ from $\overline{W}^{\star}$ is depicted in Fig. 3. All methods converge in the mean-square sense to a region around the true minimizer. The theoretical expression (42) accurately predicts the performance of the projection-free algorithm, while adding projections improves performance. Observe that notably, in this scenario, the addition of the spectral constraint to the simple constraints yields a negligible improvement, as both learning curves overlap.

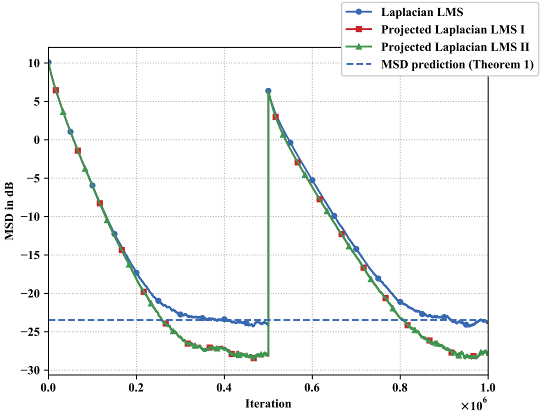

Fig. 3: Mean-Square Deviation.

6. 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] F. Chung, "The heat kernel as the pagerank of a graph," *Proceedings of the National Academy of Sciences*, vol. 104, no. 50, pp. 19735–19740, 2007.

[3] H. Ma, H. Yang, M. R. Lyu, and I. King, "Mining social networks using heat diffusion processes for marketing candidates selection," in *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, New York, NY, 2008, pp. 233–242.

[4] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 3, no. 3, pp. 484–499, Sep. 2017.

[5] A. P. Dempster, "Covariance selection," *Biometrics*, vol. 28, no. 1, pp. 157–175, 1972.

[6] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical Lasso," vol. 9, pp. 432–41, Aug. 2008.

[7] B. Lake and J. Tenenbaum, "Discovering structure by learning sparse graphs," in *Proceedings of the 32nd Annual Meeting of the Cognitive Science Society*, Jan. 2010, pp. 778–783.

[8] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst, "Learning laplacian matrix in smooth graph signal representations," *IEEE Transactions on Signal Processing*, vol. 64, no. 23, pp. 6160–6173, 2016.

[9] V. Kalofolias, "How to learn a graph from smooth signals," in *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*, Cadiz, Spain, May 2016, vol. 51, pp. 920–929.

[10] E. Pavez and A. Ortega, "Generalized laplacian precision matrix estimation for graph signal processing," in *Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Shanghai, China, Mar. 2016, pp. 6350–6354.

[11] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Transactions on Signal Processing*, vol. 65, no. 8, pp. 2077 – 2092, Apr. 2017.

[12] D. Durante and D. B. Dunson, "Locally adaptive dynamic networks," *Ann. Appl. Stat.*, vol. 10, no. 4, pp. 2203–2232, 12 2016.

[13] K. S. Xu and A. O. Hero, "Dynamic stochastic blockmodels for time-evolving social networks," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 4, pp. 552–562, Aug 2014.

[14] B. Zaman, L. M. Lopez-Ramos, D. Romero, and B. Beferull-Lozano, "Online topology estimation for vector autoregressive processes in data networks," in *IEEE 7th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, Dec 2017, pp. 1–5.

[15] F. R. K. Chung, *Spectral Graph Theory*, American Mathematical Society, 1997.

[16] K. S. Narendra and R. Shorten, "Hurwitz stability of metzler matrices," *IEEE Transactions on Automatic Control*, vol. 55, no. 6, pp. 1484–1487, June 2010.

[17] A. H. Sayed, *Adaptive Filters*, John Wiley & Sons, Inc., 2008.

[18] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of Modern Physics*, vol. 74, pp. 47–97, Jan. 2002.