Signal Processing 226 (2025) 109663

Contents lists available at ScienceDirect

Signal Processing

journal homepage: www.elsevier.com/locate/sigpro

Graph learning from incomplete graph signals: From batch to online methods

Xiang Zhang [a], Qiao Wang [a,b,*]

[a] School of Information Science and Engineering, Southeast University, Nanjing, 210096, China
[b] School of Economics and Management, Southeast University, Nanjing, 210096, China

ARTICLE INFO

Keywords:
Graph learning
Incomplete data
Graph signal recovery
Graph signal processing
Online learning

ABSTRACT

Inferring graph topologies from data is crucial in many graph-related applications. Existing works typically assume that signals are observed at all nodes, which may not hold due to application-specific constraints. The problem becomes more challenging when data are sequentially available and no delay is tolerated. To address these issues, we propose an approach for learning graphs from incomplete data. First, the problem of learning graphs with missing data is formulated as maximizing the posterior distribution with hidden variables from a Bayesian perspective. Then, we propose an expectation maximization (EM) algorithm to solve the induced problem, in which graph learning and graph signal recovery are jointly performed. Furthermore, we extend the proposed EM algorithm to an online version to accommodate the delay-sensitive situations of sequential data. Theoretically, we analyze the dynamic regret of the proposed online algorithm, illustrating the effectiveness of our algorithm in tracking graphs from partial observations in an online manner. Finally, extensive experiments on synthetic and real data are conducted, and the results corroborate that our approach can learn graphs effectively from incomplete data in both batch and online situations.

1. Introduction

Graph learning (GL) aims to reveal the graph topology underlying observed data, which can be used for downstream graph-based tasks [1,2], such as spectral clustering [3] and graph neural networks [4]. Traditional methods include inferring graphs based on the Gaussian graphical model (GGM) or Gaussian Markov random field (GMRF) model [5], structural equation model (SEM) [6], and structural vector autoregressive model (SVARM) [7]. Recently, the rise of graph signal processing (GSP) [8] provides powerful tools for learning graphs from a signal processing perspective. One of the GSP-based methods assumes that the signals are smooth over the underlying graph [9,10]. Intuitively, a graph signal being smooth means that the graph signals of two connected nodes have similar values [10]. In practice, many signals appear to be smooth such as meteorology data [9] and medical data [11], implying numerous applications of the smoothness-based methods.

It is worth noting that most existing GL works typically assume that all entities of signals are fully observed. However, we often encounter a scenario where only a subset of graph signals are available due to application-specific constraints. For example, some nodal observations in a sensor network are lost due to sensor failure or the unaffordable cost of making complete observations. Another example is social networks, where individuals are reluctant to answer some sensitive questions due to privacy concerns. Incomplete data could reduce topology inference performance significantly, thus posing a huge challenge to GL tasks.

A naive method is to directly remove the signals with missing entries and use only the complete observations to infer graph topology. However, this approach wastes available information and makes it difficult to obtain optimal results. Another approach is to first impute the missing values and use the recovered data to learn graphs via traditional GL methods. For example, the missing values can be substituted with the mean of all observed values. Recently, [12] estimates the empirical covariance matrix of incomplete data instead of raw missing data. Multiple related graphs are learned based on the estimated covariance matrix to alleviate the impact of missing data. These methods perform graph signal imputation and graph learning separately, breaking the underlying connections between the two tasks. A more reasonable strategy is to jointly recover graph signals and infer graph topology. Following this idea, [13] proposes a method that alternately estimates network processes and infers directed graphs via SEM or SVARM models. Unlike [13], the GL step of the joint schema in [14,15] is based on the smoothness assumption. Besides, [16] formulates the observation errors as non-Gaussian noise and uses the variational Bayes to learn graphs. More recently, [17,18] further consider spatiotemporal smoothness in time series. In [18], a GMRF is used to describe spatial

* Corresponding author.
E-mail address: qiaowang@seu.edu.cn (Q. Wang).

https://doi.org/10.1016/j.sigpro.2024.109663
Received 4 April 2024; Received in revised form 23 July 2024; Accepted 12 August 2024
Available online 13 August 2024

dependencies, and signals are assumed to be time-varying, which is modeled via an autoregressive (AR) model. Our model, however, is based on a GSP perspective and does not explicitly assume a dynamic signal evolution model.

However, we should mention that the above methods are of batch manner, i.e., all data are collected in advance to learn a single graph. The batch paradigm may fail to handle the cases where data are available sequentially and the delay of waiting for the graph to be fully updated is intolerable. One example is inferring social networks between users through shopping behavior. The data is only generated after each user purchase, and shopping sites hope to update the network immediately after each purchase. Existing online graph learning (OGL) methods include statistical models [19,20], causality-based [21,22] models, and smoothness-based models [23–26]. However, none of these works considers incomplete data. The first work that jointly estimates missing data and infers graph topology in an online manner is [13], which proposes a fixed-lag solver to track directed graphs based on the SVARM model. Following [13,27,28] track dynamic graphs with missing data based on the vector auto-regressive (VAR) model.

Despite the success of existing joint methods in both batch and online scenarios, they still have some problems to address. To illustrate, we summarize these methods in a unified form. To prevent confusion with subsequent notations, we let $\mathbf{Y}_o$ be the data matrix whose missing values are replaced by 0 and $\mathbf{G}$ be any matrix representing the graph topology. The joint methods [13–15,17] can be roughly summarized as

$$\min_{\mathbf{Y},\mathbf{G}} \|\mathbf{M}\circ(\mathbf{Y}-\mathbf{Y}_o)\|_{\mathrm{F}}^2 + \varpi_1 f_1(\mathbf{Y},\mathbf{G}) + \varpi_2 f_2(\mathbf{Y}) + \varpi_3 f_3(\mathbf{G}), \tag{1}$$

where ϖ_1, ϖ_2, and ϖ_3 are predefined parameters. Besides, $\mathbf{M}$ is a missing matrix whose binary values indicate whether each component is observed, and $\circ$ is Hadamard product. The first term in (1) guarantees that the recovered data $\mathbf{Y}$ does not deviate too much from $\mathbf{Y}_o$. The second term $f_1(\mathbf{Y},\mathbf{G})$ connects signals and the underlying graph via GL models, such as $f_1(\mathbf{Y},\mathbf{G}) = \mathrm{tr}(\mathbf{Y}^\top\mathbf{G}\mathbf{Y})$ for the smoothness-based GL model [14,15,17] and $f_1(\mathbf{Y},\mathbf{G}) = \|\mathbf{Y}-\mathbf{G}\mathbf{Y}\|_{\mathrm{F}}^2$ for the SEM model [13]. The terms $f_2(\mathbf{Y})$ and $f_3(\mathbf{Y})$ represent the regularizers and constraints of $\mathbf{Y}$ and $\mathbf{G}$. To solve (1), existing methods alternately update $\mathbf{Y}$ and $\mathbf{G}$, which corresponds to graph signal recovery and GL tasks, respectively. However, these joint models are heuristically designed and still lack a theoretical framework for learning graphs from partial observations. Moreover, there are many parameters in (1) (ϖ_1, ϖ_2, ϖ_3 and other parameters in the regularizers). It is exhausting to find the optimal parameters in practice. Finally, for online scenarios, none of the existing methods designs $f_1(\mathbf{Y},\mathbf{G})$ based on the GSP-based tool. Therefore, it is imperative to develop a theoretical framework to learn graphs from smooth signals in a batch and online manner without tedious parameter searches.

To fill this gap, we propose an approach for learning graphs from incomplete data based on two fundamental assumptions, i.e., signals are (i) generated from a kernel graph filter and (ii) missing (completely) at random [29]. Following the framework in [29,30], we take the missing values as hidden random variables and formulate the problem as maximizing the posterior distribution. Then, we propose an EM algorithm to solve the induced problem, in which signal recovery and graph topology inference are jointly and alternately carried out. Furthermore, we extend the batch algorithm to an online version based on the online EM framework [31] to accommodate applications with sequential data. The proposed framework is general, and we instantiate it with the widely studied smoothness-based GL model.

In summary, our contributions are summarized as follows:

- We propose a method to learn graphs with missing data from the perspective of GSP. We formulate the problem as maximizing the posterior distribution with hidden variables and propose an EM algorithm to solve the problem. The method does not need exhaustive parameter search and can be flexibly extended to many existing GL models.
- We extend the proposed batch algorithm to an online version to handle sequential data and delay-sensitive applications. Theoretically, we analyze the dynamic regret of the proposed online algorithm to reveal its ability to track dynamic graphs online from incomplete data.
- We conduct extensive experiments on synthetic and real data to validate the proposed framework, showing that our method can effectively learn graphs from incomplete data for both batch and online cases.

The rest of this paper is organized as follows. We start with background information in Section 2. The batch GL approach is presented in 3, while the online algorithm is in Section 4. Numeric tests are provided in Section 5. Finally, we make some concluding remarks in Section 6.

Notations: Throughout this paper, vectors and matrices are written in bold lowercase letters and bold uppercase letters, respectively. Given a vector $\mathbf{a}$ and a matrix $\mathbf{A}$, $\mathbf{a}[i]$ and $\mathbf{A}[ij]$ are the ith entry of $\mathbf{a}$ and the (i,j) entry of $\mathbf{A}$. Given a vector $\mathbf{a}$ or a matrix $\mathbf{A}$, $\mathbf{a}\geq 0$ ($\mathbf{A}\geq 0$) means that all elements in $\mathbf{a}$ ($\mathbf{A}$) are non-negative. Besides, $\mathbf{1}$, $\mathbf{0}$, and $\mathbf{I}$ represent all-one vectors, all-zero vectors, and identity matrices, respectively. For a vector or matrix, the ℓ_1, ℓ_2, and Frobenius norm are represented by $\|\cdot\|_1$, $\|\cdot\|_2$, $\|\cdot\|_{\mathrm{F}}$, respectively. Moreover, $\|\cdot\|_{\mathrm{F,off}}$ denotes the Frobenius norm of off-diagonal elements of a matrix, and $\mathrm{diag}(\cdot)$ means converting a vector to a diagonal matrix or vice versa. The notations $\circ$, $\dagger$, and $\mathrm{tr}(\cdot)$ stand for Hadamard product, pseudo inverse, and trace operator, respectively. Finally, given a distribution p, $\mathbb{E}[\mathbf{a}|p]$ is the expectation of $\mathbf{a}$ over the distribution p.

2. GSP background

We consider undirected graphs with non-negative weights and no self-loops. For such a graph $\mathcal{G}=\{\mathcal{V},\mathcal{E}\}$ with d vertices, where $\mathcal{V}$ and $\mathcal{E}$ are the sets of vertices and edges, respectively, its adjacency matrix $\mathbf{W}$ is a d-dimensional symmetric matrix with zero diagonal entries and non-negative off-diagonal entries. The Laplacian matrix of $\mathcal{G}$ is $\mathbf{L}=\mathbf{D}-\mathbf{W}$ [32], where the degree matrix $\mathbf{D}$ is a diagonal matrix satisfying $\mathbf{D}[ii]=\sum_{j=1}^{d}\mathbf{W}[ij]$. We study the graph signal $\mathbf{x}=[\mathbf{x}[1],\dots,\mathbf{x}[d]]^\top\in\mathbb{R}^d$ associated with $\mathcal{G}$, where $\mathbf{x}[i]$ is the signal value of node $i\in\mathcal{V}$. Given a graph signal $\mathbf{x}$ generated from the graph $\mathcal{G}$, its smoothness can be measured via [9]

$$\mathbf{x}^\top\mathbf{L}\mathbf{x} = \frac{1}{2}\sum_{i,j=1}^{d}\mathbf{W}[ij]\,(\mathbf{x}[i]-\mathbf{x}[j])^2\,. \tag{2}$$

It is observed from (2) that signal differences will be penalized more for two vertices connected by a strong edge. A small value of (2) indicates limited signal variability, meaning that $\mathbf{x}$ is smooth over the underlying graph [9].

Indeed, smooth signals can also be explained from a graph spectral perspective. Specifically, for a graph $\mathcal{G}$, its Laplacian matrix $\mathbf{L}$ has the eigen-decomposition $\mathbf{L}=\mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$, where $\mathbf{U}$ is the matrix of eigenvectors, while $\mathbf{\Lambda}$ is the diagonal matrix of eigenvalues. In the Fourier transform of GSP, eigenvalues of $\mathbf{L}$ take up a notion associated with frequency. From the graph spectral theory, a graph signal can be defined via the kernel graph as $\mathbf{x}=\boldsymbol{\mu}+g(\mathbf{L})\mathbf{n}$, where $g(\mathbf{L}):\mathbb{R}^{d\times d}\to\mathbb{R}^{d\times d}$ is a graph filter, and $\mathbf{n}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$ is the Gaussian white noise. Thus, the kernel signal will follow:

$$\mathbf{x}=\boldsymbol{\mu}+g(\mathbf{L})\mathbf{n}\sim\mathcal{N}(\boldsymbol{\mu},g^2(\mathbf{L})). \tag{3}$$

A smooth signal can be viewed as a special case of (3), where $g(\mathbf{L})$ is selected as $\sqrt{\mathbf{L}^\dagger}$. In this case, the minimizer of (2) w.r.t. $\mathbf{L}$ is a maximum likelihood estimator for $\mathbf{L}$ [33]. According to [9], if $g(\mathbf{L})=\sqrt{\mathbf{L}^\dagger}$, (3) can be further rephrased via a latent variable $\mathbf{h}\sim\mathcal{N}(\mathbf{0},\mathbf{\Lambda}^\dagger)$, i.e., $\mathbf{x}=\boldsymbol{\mu}+\mathbf{U}\mathbf{h}$. Note that $\mathbf{h}$ is inversely proportional to frequency. Thus, the corresponding graph signal $\mathbf{x}$ mainly contains the low frequency and

is smooth w.r.t. $\mathcal{G}$. We can check that the generated signals from this model follow

$$\mathbf{x} = \boldsymbol{\mu} + \mathbf{U}\mathbf{h} \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{U}\boldsymbol{\Lambda}^{\dagger}\mathbf{U}^{\top}) = \mathcal{N}(\boldsymbol{\mu}, \mathbf{L}^{\dagger}). \tag{4}$$

This signal model is of significance in the GL literature. Thus, we will focus on this model later and use it to instantiate the proposed general framework.

3. Batch graph learning from incomplete data

Given T graph signals $\mathbf{X} = [\mathbf{x}_1, \dots, \mathbf{x}_T] \in \mathbb{R}^{d\times T}$, GL aims to infer the underlying graph topology, which is encoded in the Laplacian matrix $\mathbf{L}$ or adjacent matrix $\mathbf{W}$, based on some priors on the graph topology. However, in application-specific scenarios, the graph signal $\mathbf{x}_t$ may have missing values. We use $\mathbf{x}_{t,o}$ and $\mathbf{x}_{t,m}$ to denote the observed parts and the missing parts of $\mathbf{x}_t$, respectively. In the case of incomplete data, we learn graphs only using the observed signals, $\mathbf{x}_{1,o}, \dots, \mathbf{x}_{T,o}$. In this section, we learn graphs from batch incomplete data, where all T incomplete data $\mathbf{x}_{1,o}, \dots, \mathbf{x}_{T,o}$ are collected in advance and used to learn an optimal graph.

3.1. Model formulation

For clarity, we define $\mathbf{X}_o = \{\mathbf{x}_{t,o}\}_{t=1}^T$ and $\mathbf{X}_m = \{\mathbf{x}_{t,m}\}_{t=1}^T$. Furthermore, consistent with (1), the missing matrix $\mathbf{M} \in \mathbb{R}^{D\times T}$ is a binary matrix whose binary values indicate whether the corresponding entry is observed. Inspired by [30], $\mathbf{M}$ is considered to be a random variable. In practice, we can observe $\mathbf{M}$ and $\mathbf{X}_o$, and the joint density of observations is

$$p(\mathbf{X}_o, \mathbf{M}|\boldsymbol{\mu}, \mathbf{L}, \boldsymbol{\nu}) = \int p(\mathbf{X}_o, \mathbf{X}_m|\boldsymbol{\mu}, \mathbf{L}) p(\mathbf{M}|\mathbf{X}_o, \mathbf{X}_m, \boldsymbol{\nu}) d\mathbf{X}_m, \tag{5}$$

where $p(\mathbf{X}_o, \mathbf{X}_m|\boldsymbol{\mu}, \mathbf{L})$ is the generation model of the complete graph signals, $p(\mathbf{M}|\mathbf{X}_o, \mathbf{X}_m, \boldsymbol{\nu})$ defines the missing mechanism, and $\boldsymbol{\nu}$ is any parameter vector of the missing mechanism, which is assumed to be independent of $\boldsymbol{\mu}$ and $\mathbf{L}$. The missing mechanisms can be divided into three categories according to whether $p(\mathbf{M}|\mathbf{X}_o, \mathbf{X}_m, \boldsymbol{\nu})$ is independent of $\mathbf{X}_o$ and $\mathbf{X}_m$ [29]. Specifically,

- if $p(\mathbf{M}|\mathbf{X}_o, \mathbf{X}_m, \boldsymbol{\nu}) = p(\mathbf{M}|\boldsymbol{\nu})$, the missing data are missing completely at random (MCAR);
- if $p(\mathbf{M}|\mathbf{X}_o, \mathbf{X}_m, \boldsymbol{\nu}) = p(\mathbf{M}|\mathbf{X}_o, \boldsymbol{\nu})$, the missing data are missing at random (MAR);
- if $p(\mathbf{M}|\mathbf{X}_o, \mathbf{X}_m, \boldsymbol{\nu})$ depends on the missing data $\mathbf{X}_m$, the missing data are missing not at random (MNAR).

To build our approach, we make the following assumptions.

Assumption 1. The complete data $\mathbf{x}_t$ follows the distribution $\mathcal{N}(\boldsymbol{\mu}, g^2(\mathbf{L})), \forall t$.

Assumption 2. The data are missing (completely) at random, i.e., the missing mechanisms are MCAR or MAR.

Assumption 1 is common in the GL literature and connects the signals to the underlying graph from the perspective of GSP. As mentioned before, the widely studied smoothness-based GL models [9,10] are special cases of this signal model where $g(\mathbf{L})$ is selected as $\sqrt{\mathbf{L}^{\dagger}}$. Assumption 2 defines the missing data mechanism of our model and means that the missing entries are independent of missing data. This assumption is widely used in model learning with missing data [12,34]. Based on the assumptions, given the observations $\mathbf{X}_o$ and $\mathbf{M}$, we learn graphs from incomplete data by maximizing the posterior distribution $p(\boldsymbol{\mu}, \mathbf{L}, \boldsymbol{\nu}|\mathbf{X}_o, \mathbf{M})$ w.r.t. $\boldsymbol{\mu}$ and $\mathbf{L}$. We have

$$\begin{aligned}
&\operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \; L(\boldsymbol{\mu}, \mathbf{L}; \mathbf{X}_o, \mathbf{M}) && (6)\\
:= &\operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \; \ln p(\boldsymbol{\mu}, \mathbf{L}, \boldsymbol{\nu}|\mathbf{X}_o, \mathbf{M}) && (7)\\
\propto &\operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \; \ln p(\mathbf{X}_o, \mathbf{M}|\boldsymbol{\mu}, \mathbf{L}, \boldsymbol{\nu}) p(\mathbf{L}) p(\boldsymbol{\mu}) p(\boldsymbol{\nu}) && (8)\\
= &\operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \; \ln \int p(\mathbf{X}_o, \mathbf{X}_m|\boldsymbol{\mu}, \mathbf{L}) p(\mathbf{M}|\mathbf{X}_o, \mathbf{X}_m, \boldsymbol{\nu}) p(\mathbf{L}) d\mathbf{X}_m && (9)\\
= &\operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \; \ln \int p(\mathbf{X}_o, \mathbf{X}_m|\boldsymbol{\mu}, \mathbf{L}) p(\mathbf{L}) d\mathbf{X}_m && (10)\\
:= &\operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \; L(\boldsymbol{\mu}, \mathbf{L}; \mathbf{X}_o) && (11)\\
= &\operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \; \ln \prod_{t=1}^{T} \int p(\mathbf{x}_{t,o}, \mathbf{x}_{t,m}|\boldsymbol{\mu}, \mathbf{L}) p(\mathbf{L}) d\mathbf{x}_{t,m} && (12)\\
= &\operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \; \sum_{t=1}^{T} \ln \int p(\mathbf{x}_{t,o}, \mathbf{x}_{t,m}|\boldsymbol{\mu}, \mathbf{L}) p(\mathbf{L}) d\mathbf{x}_{t,m} && (13)\\
= &\operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \; \sum_{t=1}^{T} \ln \int \mathcal{N}(\mathbf{x}_t : \boldsymbol{\mu}, \boldsymbol{\Sigma}) p(\mathbf{L}) d\mathbf{x}_{t,m}, && (14)
\end{aligned}$$

where $\boldsymbol{\Sigma} := g^2(\mathbf{L})$ and $\mathcal{L}$ is set containing all possible Laplacian matrices, i.e., $\mathcal{L} := \{\mathbf{L} : \mathbf{L}\mathbf{1} = \mathbf{0}, \mathbf{L}[ij] = \mathbf{L}[ji] < 0, i \neq j\}$. Note that $p(\mathbf{L})$ encodes the priors on the graph topology. Furthermore, (9) holds since we assume no prior is available for $\boldsymbol{\mu}$, and $\boldsymbol{\nu}$ is not the optimization variable. Based on Assumption 2, $p(\mathbf{M}|\mathbf{X}_o, \mathbf{X}_m, \boldsymbol{\nu})$ is independent of $\mathbf{X}_m$, and (10) hence holds. Eq. (12) holds since the samples are i.i.d. Finally, (14) holds due to Assumption 1. Although we have the explicit form of $\mathcal{N}(\mathbf{x}_t : \boldsymbol{\mu}, \boldsymbol{\Sigma})$, the integral in (14) makes the problem difficult to solve. Thus, we propose an EM algorithm to solve the problem.

Remark 1. To the best of our knowledge, our approach is the first work that considers missing mechanisms and models the problem of learning graphs with missing data from a Bayesian perspective. The work [12] assumes the missing mechanism is MCAR, but it is a deterministic approach. Besides, the work [16] models the observational errors as non-Gaussian noises using probabilistic tools. However, it does not consider missing mechanisms. Furthermore, our model differs from [16] in that we formulate missing values as latent variables and use the EM algorithm to solve the problem.

3.2. Proposed batch EM algorithm

The proposed algorithm consists of two steps, i.e., the E step and the M step. In the E step, we estimate the expected complete data given the observed parts. In the M step, we optimize $\boldsymbol{\mu}$ and $\mathbf{L}$ based on the estimated complete data.

3.2.1. E step

Following the standard EM framework, given $\boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}$ estimated from the kth iteration, we need to calculate the Q function in the E step, i.e., $Q(\boldsymbol{\mu}, \mathbf{L}; \boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}) := \mathbb{E}\left[\ln p(\mathbf{X}_o, \mathbf{X}_m|\boldsymbol{\mu}, \mathbf{L}) p(\mathbf{L})|\widetilde{p}(\mathbf{X})\right]$, where we define $\widetilde{p}(\mathbf{X}) = p(\mathbf{X}|\boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}, \mathbf{X}_o)$. Here, since all data are sampled independently, we have $Q(\boldsymbol{\mu}, \mathbf{L}; \boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}) = \sum_{t=1}^{T} \mathbb{E}\left[\ln p(\mathbf{x}_{t,o}, \mathbf{x}_{t,m}|\boldsymbol{\mu}, \mathbf{L}) + \ln p(\mathbf{L})|\widetilde{p}(\mathbf{x}_t)\right]$, where $\widetilde{p}(\mathbf{x}_t) := p\left(\mathbf{x}_t|\boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}, \mathbf{x}_{t,o}\right)$. By definition, the key to the Q function is to compute the expected complete data $\mathbf{x}_t$ based on the observed data $\mathbf{x}_{t,o}$. Without loss of generality, we let $\mathbf{x}_t = [\mathbf{x}_{t,o}^{\top}, \mathbf{x}_{t,m}^{\top}]^{\top}$, and the corresponding $\boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)}$ can be decomposed into:

$$\boldsymbol{\mu}^{(k)} = \begin{bmatrix} \boldsymbol{\mu}_o^{(k)} \\ \boldsymbol{\mu}_m^{(k)} \end{bmatrix}, \boldsymbol{\Sigma}^{(k)} = \begin{bmatrix} \boldsymbol{\Sigma}_{oo}^{(k)} & \boldsymbol{\Sigma}_{om}^{(k)} \\ \boldsymbol{\Sigma}_{mo}^{(k)} & \boldsymbol{\Sigma}_{mm}^{(k)} \end{bmatrix}. \tag{15}$$

Note that $p\left(\mathbf{x}_t|\boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}\right) = \mathcal{N}\left(\mathbf{x}_t : \boldsymbol{\mu}^{(k)}, \boldsymbol{\Sigma}^{(k)}\right)$, and the conditional distribution $p\left(\mathbf{x}_{t,m}|\boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}, \mathbf{x}_{t,o}\right)$ also follows Gaussian distribution [35]. Let $\boldsymbol{\xi}_t \in \mathbb{R}^d$ and $\boldsymbol{\Xi}_t \in \mathbb{R}^{d\times d}$ be the first and second order sufficient statistics of $\mathbf{x}_t$ given $\mathbf{x}_{t,o}$. It is not difficult to obtain that [30]

$$\boldsymbol{\xi}_t^{(k+1)} = \mathbb{E}\left[\mathbf{x}_t|\widetilde{p}(\mathbf{x}_t)\right] = \begin{bmatrix} \boldsymbol{\xi}_{t,o}^{(k+1)} \\ \boldsymbol{\xi}_{t,m}^{(k+1)} \end{bmatrix}, \tag{16}$$

$$\boldsymbol{\Xi}_t^{(k+1)} = \mathbb{E}\left[\mathbf{x}_t\mathbf{x}_t^{\top}|\widetilde{p}(\mathbf{x}_t)\right] = \begin{bmatrix} \boldsymbol{\Xi}_{t,oo}^{(k+1)} & \boldsymbol{\Xi}_{t,om}^{(k+1)} \\ \boldsymbol{\Xi}_{t,mo}^{(k+1)} & \boldsymbol{\Xi}_{t,mm}^{(k+1)} \end{bmatrix}, \tag{17}$$

where

$$\xi_{t,o}^{(k+1)} = \mathbf{x}_{t,o},\quad \xi_{t,m}^{(k+1)} = \boldsymbol{\mu}_m^{(k)} + \boldsymbol{\Sigma}_{mo}^{(k)}\left(\boldsymbol{\Sigma}_{oo}^{(k)}\right)^{-1}(\mathbf{x}_{t,o} - \boldsymbol{\mu}_o^{(k)}), \tag{18}$$

$$\Xi_{t,oo}^{(k+1)} = \xi_{t,o}^{(k+1)}\left(\xi_{t,o}^{(k+1)}\right)^{\top},\quad \Xi_{t,mo}^{(k+1)} = \xi_{t,m}^{(k+1)}\left(\xi_{t,o}^{(k+1)}\right)^{\top}, \Xi_{t,om}^{(k+1)} = \xi_{t,o}^{(k+1)}\left(\xi_{t,m}^{(k+1)}\right)^{\top},$$

$$\Xi_{t,mm}^{(k+1)} = \boldsymbol{\Sigma}_{mm}^{(k)} - \boldsymbol{\Sigma}_{mo}^{(k)}\left(\boldsymbol{\Sigma}_{oo}^{(k)}\right)^{-1}\boldsymbol{\Sigma}_{om}^{(k)} + \xi_{t,m}^{(k+1)}\left(\xi_{t,m}^{(k+1)}\right)^{\top}. \tag{19}$$

Indeed, the updates of $\xi_{t,m}^{(k+1)}$ and $\Xi_{t,mm}^{(k+1)}$ are the solutions of linear regressions, which can be efficiently solved via methods such as the sweep operator [29].

3.2.2. M step

After obtaining $\xi_t^{(k+1)}$ and $\Xi_t^{(k+1)}$, we can infer $\boldsymbol{\mu}^{(k+1)}$ and $\mathbf{L}^{(k+1)}$ via

$$\begin{aligned}\boldsymbol{\mu}^{(k+1)}, \mathbf{L}^{(k+1)} &= \operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} Q(\boldsymbol{\mu}, \mathbf{L}; \boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)})\\ &= \operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \sum_{t=1}^{T}\mathbb{E}\left[\ln p(\mathbf{x}_{t,o}, \mathbf{x}_{t,m}|\boldsymbol{\mu}, \mathbf{L}) + \ln p(\mathbf{L})|\widetilde{p}(\mathbf{x}_t)\right]\\ &= \operatorname*{argmax}_{\boldsymbol{\mu},\mathbf{L}\in\mathcal{L}} \sum_{t=1}^{T}\mathbb{E}\left[\ln \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}, \boldsymbol{\Sigma}) + \ln p(\mathbf{L})|\widetilde{p}(\mathbf{x}_t)\right].\end{aligned} \tag{20}$$

The problem (20) has two subproblems, i.e., updating $\boldsymbol{\mu}$ and $\mathbf{L}$.

Updating $\boldsymbol{\mu}^{(k+1)}$: The problem of updating $\boldsymbol{\mu}^{(k+1)}$ is

$$\begin{aligned}\boldsymbol{\mu}^{(k+1)} &= \operatorname*{argmax}_{\boldsymbol{\mu}} \sum_{t=1}^{T}\mathbb{E}\left[\ln \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}, \mathbf{L})|\widetilde{p}(\mathbf{x}_t)\right]\\ &= \operatorname*{argmin}_{\boldsymbol{\mu}} \sum_{t=1}^{T}\mathbb{E}\left[\frac{(\mathbf{x}_t - \boldsymbol{\mu})^{\top}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_t - \boldsymbol{\mu})}{2}\middle|\widetilde{p}(\mathbf{x}_t)\right]\\ &= \operatorname*{argmin}_{\boldsymbol{\mu}} \sum_{t=1}^{T}\boldsymbol{\mu}^{\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - 2\mathbb{E}\left[\mathbf{x}_t^{\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}|\widetilde{p}(\mathbf{x}_t)\right]\\ &:= \operatorname*{argmin}_{\boldsymbol{\mu}} \psi(\boldsymbol{\mu}).\end{aligned} \tag{21}$$

Taking the derivative of $\psi(\boldsymbol{\mu})$ and setting to zero, we have

$$\boldsymbol{\mu}^{(k+1)} = \frac{\sum_{t=1}^{T}\mathbb{E}\left[\mathbf{x}_t|\widetilde{p}(\mathbf{x}_t)\right]}{T} = \frac{\sum_{t=1}^{T}\xi_t^{(k+1)}}{T}. \tag{22}$$

Updating $\mathbf{L}^{(k+1)}$: The problem of updating $\mathbf{L}^{(k+1)}$ is

$$\begin{aligned}\mathbf{L}^{(k+1)} &= \operatorname*{argmax}_{\mathbf{L}\in\mathcal{L}} \sum_{t=1}^{T}\mathbb{E}\left[\ln \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}, \mathbf{L}) + \ln p(\mathbf{L})|\widetilde{p}(\mathbf{x}_t)\right]\\ &\approx \operatorname*{argmin}_{\mathbf{L}\in\mathcal{L}} \sum_{t=1}^{T}\boldsymbol{\mu}^{\top}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - 2\mathbb{E}\left[\mathbf{x}_t^{\top}|\widetilde{p}(\mathbf{x}_t)\right]\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\\ &\quad + \mathbb{E}\left[\mathbf{x}_t^{\top}\mathbf{x}_t|\widetilde{p}(\mathbf{x}_t)\right]\boldsymbol{\Sigma}^{-1} + r(\mathbf{L})\end{aligned} \tag{23a}$$

$$\begin{aligned}&= \operatorname*{argmin}_{\mathbf{L}\in\mathcal{L}} T\operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\boldsymbol{\mu}^{\top}\right) - 2\operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}\sum_{t=1}^{T}\mathbb{E}[\mathbf{x}_t^{\top}|\widetilde{p}(\mathbf{x}_t)]\right)\\ &\quad + \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\sum_{t=1}^{T}\mathbb{E}[\mathbf{x}_t\mathbf{x}_t^{\top}|\widetilde{p}(\mathbf{x}_t)]\right) + Tr(\mathbf{L})\\ &= \operatorname*{argmin}_{\mathbf{L}\in\mathcal{L}} \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\sum_{t=1}^{T}\Xi_t^{(k+1)}\right) + Tr(\mathbf{L})\\ &\quad - T\operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}^{(k+1)}\left(\boldsymbol{\mu}^{(k+1)}\right)^{\top}\right)\end{aligned} \tag{23b}$$

$$= \operatorname*{argmin}_{\mathbf{L}\in\mathcal{L}} \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\mathbf{Q}\right) + r(\mathbf{L}), \tag{23c}$$

where $\mathbf{Q} = \frac{\sum_{t=1}^{T}\Xi_t^{(k+1)}}{T} - \boldsymbol{\mu}^{(k+1)}\left(\boldsymbol{\mu}^{(k+1)}\right)^{\top}$. Moreover, (23b) holds due to (17) and (22), and $r(\mathbf{L}) = \ln p(\mathbf{L})$ is the graph prior encoded in the $p(\mathbf{L})$. We omit the $\ln\det(\boldsymbol{\Sigma})$ term in the log-likelihood of Gaussian distributions in (23a) for the following reasons, where $\det(\cdot)$ is the (pseudo)determinant of a matrix. First, this is inspired by [33], which states that the induced topology inference problem (23c) exhibits more interpretability and accuracy than the origin covariance estimation problem. Second, we care about a general GL framework, where the origin problem is a special case of ours since $\ln\det(\boldsymbol{\Sigma})$ can be incorporated into $r(\mathbf{L})$. Third, using a well-designed $r(\mathbf{L})$ instead of $\ln\det(\boldsymbol{\Sigma})$ can simplify the problem and lead to better results.

Algorithm 1 Btach EM algorithms for graph learning from incomplete data (BEMGLID)

Input:
The observed data $\mathbf{X}_o$
Output:
The learned graph $\mathbf{L}$ and the recovered signals $\mathbf{X}$
Initialize $\boldsymbol{\mu}^{(0)} = \mathbf{0}$, $\mathbf{w}^{(0)} \geq 0$ randomly, and let $\mathbf{L}^{(0)} = \mathcal{T}\mathbf{w}^{(0)}$
for $k = 0, ..., K-1$ **do**
  E step:
  for $t = 1, ..., T$ **do**
    Update $\xi_t^{(k+1)}$ and $\Xi_t^{(k+1)}$ using (16) and (17), respectively
  end for
  M step:
  Update $\boldsymbol{\mu}^{(k+1)}$ via (22)
  Update $\mathbf{w}^{(k+1)}$ by solving (24) using PDS algorithm is [10]
  Let $\mathbf{L}^{(k+1)} = \mathcal{T}\mathbf{w}^{(k+1)}$
end for

Note that (23c) is a general model that can be applied to many existing GL methods. Here, we take the widely studied smoothness-based GL model [10] as an example to explain how (23c) works. Specifically, if we let $g(\mathbf{L}) = \sqrt{\mathbf{L}^{\dagger}}$, which is a low-pass graph filter, $\operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\mathbf{Q}\right)$ in (23c) is $\operatorname{tr}(\mathbf{LQ})$. Besides, the model in [10] chooses $p(\mathbf{L}) \propto \exp(\alpha\mathbf{1}^{\top}\ln(\operatorname{diag}(\mathbf{L})) - \beta\|\mathbf{L}\|_{\mathrm{F,off}}^2)$, where α and β are predefined parameters. Thus, problem (23c) is equivalent to

$$\operatorname*{argmin}_{\mathbf{L}\in\mathcal{L}} \operatorname{tr}(\mathbf{LQ}) - \alpha\mathbf{1}^{\top}\ln(\operatorname{diag}(\mathbf{L})) + \beta\|\mathbf{L}\|_{\mathrm{F,off}}^2, \tag{24}$$

where the second term in (24) is used to control node connectivity, while the third term is to control sparsity. The problem is convex and can be solved efficiently by many methods [10,36]. Here, we use the primal–dual split (PDS) algorithm in [10] to solve (24). Note that the number of free variables of $\mathbf{L}$ is $s = \frac{d(d-1)}{2}$ due to the definition of $\mathcal{L}$. Hence, we define a vector $\mathbf{w} \in \mathbb{R}^s$ that contains the upper triangle elements of $\mathbf{W}$. The vector can be transformed into a Laplacian matrix via an operator $\mathcal{T} : \mathbb{R}^s \to \mathbb{R}^{d\times d}$, i.e., $\mathbf{L} = \mathcal{T}\mathbf{w}$ [37]. Besides, $\mathcal{T}^* : \mathbb{R}^{d\times d} \to \mathbb{R}^s$ is the adjoint operator of $\mathcal{T}$. The definitions of $\mathcal{T}$ and $\mathcal{T}^*$ are placed in Appendix A. Then, the problem (24) can be rephrased as

$$\begin{aligned}\operatorname{argmin}\varphi(\mathbf{w}) &:= \operatorname{argmin}\operatorname{tr}(\mathcal{T}\mathbf{wQ}) - \alpha\mathbf{1}^{\top}\ln(\mathbf{Sw}) + 2\beta\|\mathbf{w}\|_2^2 + \operatorname{Ind}(\mathbf{w})\\ &= \operatorname{argmin}\langle\mathbf{w}, \mathcal{T}^*\mathbf{Q}\rangle - \alpha\mathbf{1}^{\top}\ln(\mathbf{Sw}) + 2\beta\|\mathbf{w}\|_2^2 + \operatorname{Ind}(\mathbf{w})\end{aligned} \tag{25}$$

where $\mathbf{S} \in \mathbb{R}^{d\times s}$ is a matrix satisfying $\mathbf{Sw} = \mathbf{W1}$, and $\operatorname{Ind}(\cdot)$ is defined as

$$\operatorname{Ind}(\mathbf{w}) = \begin{cases}0 & \mathbf{w} \geq 0,\\ +\infty & \text{else.}\end{cases} \tag{26}$$

The problem (25) is of the same form as that in [10]. Thus, it can be solved via the algorithm in [10]. Suppose the solution of (25) is $\hat{\mathbf{w}}$, we let $\mathbf{L}^{(k+1)} = \mathcal{T}\hat{\mathbf{w}}$.

The overall algorithm is named as **B**atch **EM** algorithm for **G**raph **L**earning from **I**ncomplete **D**ata (BEMGLID), which is placed in Algorithm 1. The computational cost of our algorithm consists of two parts, i.e., signal recovery and graph learning. The signal recovery step needs $\mathcal{O}(d_o^3 + d_m^2 d_o)$, where d_o is the number of visible nodes and d_m is the number of invisible nodes. However, the signal recovery step can be viewed as a linear regression solution, which can be efficiently solved via methods such as the sweep operator to reduce complexity. For the graph learning step, the algorithm we used [10] needs $\mathcal{O}(d^2)$ cost. In the literature, the joint models of learning graphs from incomplete data need $\mathcal{O}(d^3)$ [13–15,17,18]. Therefore, our algorithm can achieve comparable computational costs with existing algorithms.

3.3. Convergence analysis

First, we have

$$p(\mathbf{X}_o, \mathbf{X}_m|\boldsymbol{\mu}, \mathbf{L}) = p(\mathbf{X}_o|\boldsymbol{\mu}, \mathbf{L})p(\mathbf{X}_m|\mathbf{X}_o, \boldsymbol{\mu}, \mathbf{L}). \tag{27}$$

Taking $\ln(\cdot)$ and adding $\ln p(\mathbf{L})$ on both sides of (27), we can obtain

$$L(\boldsymbol{\mu}, \mathbf{L}; \mathbf{X}_o, \mathbf{X}_m) = L(\boldsymbol{\mu}, \mathbf{L}; \mathbf{X}_o) + \ln p(\mathbf{X}_m|\mathbf{X}_o, \boldsymbol{\mu}, \mathbf{L}). \tag{28}$$

Next, we take the expectation of all terms in (28) over $\widetilde{p}(\mathbf{X})$ and have

$$L(\boldsymbol{\mu}, \mathbf{L}; \mathbf{X}_o) = Q(\boldsymbol{\mu}, \mathbf{L}; \boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}) - H(\boldsymbol{\mu}, \mathbf{L}; \boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}), \tag{29}$$

where

$$H(\boldsymbol{\mu}, \mathbf{L}; \boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}) = \int \left[\ln p(\mathbf{X}_m|\mathbf{X}_o, \boldsymbol{\mu}, \mathbf{L})\right] \widetilde{p}(\mathbf{X}) d\mathbf{X}_m. \tag{30}$$

We have $H(\boldsymbol{\mu}^{(k+1)}, \mathbf{L}^{(k+1)}; \boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}) \leq H(\boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}; \boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)})$ from Jensen's inequality. Furthermore, $\boldsymbol{\mu}^{(k+1)}$ and $\mathbf{L}^{(k+1)}$ are updated from the M step, which increases $Q(\boldsymbol{\mu}, \mathbf{L}; \boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)})$. Combining the two points, we can conclude that $L(\boldsymbol{\mu}^{(k+1)}, \mathbf{L}^{(k+1)}; \mathbf{X}_o) \leq L(\boldsymbol{\mu}^{(k)}, \mathbf{L}^{(k)}; \mathbf{X}_o)$, meaning that $\boldsymbol{\mu}^{(k+1)}, \mathbf{L}^{(k+1)}$ increases the log-likelihood in (11) after each update until at least a local maximum is reached.

3.4. Some discussions

Some discussions on the proposed method are as follows.

(1) As stated in [12], the alternating update algorithm for solving (1) of existing methods [13–15,17] can be regarded as an analogue of the EM algorithm. However, our approach differs from the above methods in the following aspects. First, the model (1) is deterministic and the corresponding algorithm is designed heuristically, while our EM algorithm is rigorously derived through a Bayesian framework. Second, our method does not require exhaustive parameter searches as in (1). Third, the E step of our graph can be viewed as graph signal recovery, i.e., impute missing graph signals from the learned graph. Existing works recover missing signals by solving an optimization problem, such as the graph total variation minimization [38–41]. It is observed from (18) that our method imputes the missing data via a linear regression based on a graph signal generation model.

(2) Our method learns a single graph from incomplete data. In the literature, [12] attempts to leverage the topological relationships among multiple graphs to alleviate the impact of missing values. The philosophy is that jointly learning multiple graphs can improve inference performance by borrowing information from the data of other graphs. Our approach can be easily extended to multiple graph learning if we appropriately choose topological priors. Specifically, suppose there are C graphs $\mathbf{L}_1, \ldots, \mathbf{L}_C$ producing incomplete data. If we let the prior distribution $p(\mathbf{L}_1, \ldots, \mathbf{L}_C) \propto \exp(\sum_{c=1}^{C} -r(\mathbf{L}_c) - \sum_{c=2}^{C} \|\mathbf{L}_c - \mathbf{L}_{c-1}\|_{1,1})$, the M step in our model is similar to that in [12]. Different from [12], our algorithm has the additional E step to recover the missing signals for each graph separately.

(3) Due to the flexibility of the Bayesian framework, our framework can be easily extended to the graph Laplacian mixture models in [33], where data are provided in a mixture form and the corresponding data labels are unknown. Specifically, by introducing a latent variable to indicate data labels, we can model the problem as a Gaussian mixture model with incomplete data. This will be left to future work.

4. Online graph learning from incomplete data

In this section, we learn graphs from the incomplete data stream, where $\mathbf{x}_{1,o}, \mathbf{x}_{2,o}, \ldots, \mathbf{x}_{T,o}$ arrive sequentially. We receive $\mathbf{x}_{t,o}$ at the tth time slot and are required to update a new graph $\mathbf{L}_t$ (or $\mathbf{W}_t$) immediately. To this end, we first extend the above batch algorithm to an online version.

4.1. Proposed online EM algorithm

Similar to the batch algorithm, the proposed online algorithm consists of two steps, i.e., the online E step and the online M step.

4.1.1. Online E step

Suppose that we have updated $\boldsymbol{\xi}_{t-1}$, $\boldsymbol{\Xi}_{t-1}$, $\boldsymbol{\mu}_{t-1}$, and $\mathbf{L}_{t-1}$ using data $\mathbf{x}_{1,o}, \ldots, \mathbf{x}_{t-1,o}$. Inspired by [31], after receiving $\mathbf{x}_{t,o}$ at the tth time slot, we calculate the following $Q_t(\boldsymbol{\mu}, \mathbf{L})$ function[1] recursively in the E step via

$$Q_t(\boldsymbol{\mu}, \mathbf{L}) = (1-\rho)Q_{t-1}(\boldsymbol{\mu}, \mathbf{L}) + \rho\mathbb{E}\left[\ln p(\mathbf{x}_t|\boldsymbol{\mu}, \mathbf{L})p(\mathbf{L})|\check{p}(\mathbf{x}_t)\right], \tag{31}$$

where $\check{p}(\mathbf{x}_t) = p(\mathbf{x}_t|\boldsymbol{\mu}_{t-1}, \mathbf{L}_{t-1}, \mathbf{x}_{t,o})$, and ρ is a predetermined forgetting factor controlling the weight of past information. Like the batch algorithm, calculating (31) is equivalent to updating $\boldsymbol{\xi}_t$ and $\boldsymbol{\Xi}_t$ via

$$\boldsymbol{\xi}_t = (1-\rho)\boldsymbol{\xi}_{t-1} + \rho\widetilde{\boldsymbol{\xi}}_t \tag{32}$$

$$\boldsymbol{\Xi}_t = (1-\rho)\boldsymbol{\Xi}_{t-1} + \rho\widetilde{\boldsymbol{\Xi}}_t, \tag{33}$$

where

$$\begin{aligned}
&\widetilde{\boldsymbol{\xi}}_t = \mathbb{E}\left[\mathbf{x}_t|\check{p}(\mathbf{x}_t)\right] = \begin{bmatrix}\widetilde{\boldsymbol{\xi}}_{t,o}\\ \widetilde{\boldsymbol{\xi}}_{t,m}\end{bmatrix}, \quad \widetilde{\boldsymbol{\Xi}}_t = \mathbb{E}\left[\mathbf{x}_t\mathbf{x}_t^\top|\check{p}(\mathbf{x}_t)\right] = \begin{bmatrix}\widetilde{\boldsymbol{\Xi}}_{t,oo} & \widetilde{\boldsymbol{\Xi}}_{t,om}\\ \widetilde{\boldsymbol{\Xi}}_{t,mo} & \widetilde{\boldsymbol{\Xi}}_{t,mm}\end{bmatrix},\\
&\widetilde{\boldsymbol{\xi}}_{t,o} = \mathbf{x}_{t,o}, \quad \widetilde{\boldsymbol{\xi}}_{t,m} = \boldsymbol{\mu}_{t-1,m} + \boldsymbol{\Sigma}_{t-1,mo}\left(\boldsymbol{\Sigma}_{t-1,oo}\right)^{-1}(\mathbf{x}_{t,o} - \boldsymbol{\mu}_{t-1,o}),\\
&\widetilde{\boldsymbol{\Xi}}_{t,oo} = \widetilde{\boldsymbol{\xi}}_{t,o}\left(\widetilde{\boldsymbol{\xi}}_{t,o}\right)^\top, \quad \widetilde{\boldsymbol{\xi}}_{t,mo} = \widetilde{\boldsymbol{\xi}}_{t,m}\left(\widetilde{\boldsymbol{\xi}}_{t,o}\right)^\top, \quad \widetilde{\boldsymbol{\Xi}}_{t,om} = \widetilde{\boldsymbol{\xi}}_{t,o}\left(\widetilde{\boldsymbol{\xi}}_{t,m}\right)^\top,\\
&\widetilde{\boldsymbol{\Xi}}_{t,mm} = \boldsymbol{\Sigma}_{t-1,mm} - \boldsymbol{\Sigma}_{t-1,mo}\left(\boldsymbol{\Sigma}_{t-1,oo}\right)^{-1}\boldsymbol{\Sigma}_{t-1,om} + \widetilde{\boldsymbol{\xi}}_{t,m}\left(\widetilde{\boldsymbol{\xi}}_{t,m}\right)^\top.
\end{aligned} \tag{34}$$

The merits of this recursive schema are three-fold. First, we only need to store $\boldsymbol{\xi}_{t-1}$ and $\boldsymbol{\Xi}_{t-1}$ instead of all obtained data, which saves storage space. Second, the computational burden can be reduced since we only need to recover $\widetilde{\boldsymbol{\xi}}_t$ and $\widetilde{\boldsymbol{\Xi}}_t$ at t. Third, removing past information via ρ can improve the performance of tracking dynamic graphs.

4.1.2. Online M step

After obtaining $\boldsymbol{\xi}_t$ and $\boldsymbol{\Xi}_t$, similar to the M step of the batch algorithm, we maximize the Q function (31) to update $\boldsymbol{\mu}_t$ and $\mathbf{L}_t$, i.e.,

$$\begin{aligned}
\boldsymbol{\mu}_t, \mathbf{L}_t &= \operatorname*{argmax}_{\boldsymbol{\mu}, \mathbf{L}\in\mathcal{L}} Q_t(\boldsymbol{\mu}, \mathbf{L})\\
&= \rho\mathbb{E}\left[\ln \mathcal{N}(\mathbf{x}_t; \boldsymbol{\mu}, \mathbf{L}) + \ln p(\mathbf{L})|\check{p}(\mathbf{x}_t)\right] + (1-\rho)Q_{t-1}(\boldsymbol{\mu}, \mathbf{L}).
\end{aligned} \tag{35}$$

The problem (35) has two subproblems, which will be explained next.

Updating $\boldsymbol{\mu}_t$: By the same method of solving (21), after some calculations, we have

$$\boldsymbol{\mu}_t = (1-\rho)\boldsymbol{\xi}_{t-1} + \rho\widetilde{\boldsymbol{\xi}}_t = \boldsymbol{\xi}_t. \tag{36}$$

Updating $\mathbf{L}_t$: Following (23c), we can update $\mathbf{L}_t$ via

$$\mathbf{L}_t = \operatorname*{argmin}_{\mathbf{L}\in\mathcal{L}} \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\mathbf{Q}_t\right) + r(\mathbf{L}), \tag{37}$$

where $\mathbf{Q}_t = \boldsymbol{\Xi}_t - \boldsymbol{\mu}_t\boldsymbol{\mu}_t^\top$. The problem is similar to (23c) and therefore can be solved by many algorithms as in the batch scenario. However, in delay-sensitive applications, we may not be able to solve (37) exactly due to computational and time constraints. According to existing OGL methods [42,43], it is more reasonable to employ the online gradient descent type algorithm, which makes only one forward step at each time slot to handle delay-sensitive constraints. Here, we use the inexact proximal online gradient descent (IP-OGD) algorithm [44] since the estimated $\mathbf{Q}_t$ is inexact due to missing values. We still take the smoothness-based model (24) as an example. At the beginning of

[1] Unlike the previous section, we omit the parameters $\boldsymbol{\mu}$ and $\mathbf{L}$ in the Q function for the sake of notational simplicity.

Algorithm 2 Online EM algorithm for graph learning from incomplete data (OEMGLID)

Input:
The forgetting factor ρ and stepsize η
Output:
The learned graph $\mathbf{L}_t$ and recovered signal $\mathbf{x}_t$, $t = 1, ..., T$
Initialize $\boldsymbol{\mu}_0 = \mathbf{0}, \widetilde{\boldsymbol{\xi}}_0 = \mathbf{0}, \widetilde{\boldsymbol{\Xi}}_0 = \mathbf{0}$, $\mathbf{w}_0 \geq 0$ randomly, and let $\mathbf{L}_0 = \mathcal{T}\mathbf{w}_0$,
for $t = 1, ..., T$ **do**
  Receive data $\mathbf{x}_{t,o}$ sequentially
  E step:
  Update $\widetilde{\boldsymbol{\xi}}$ and $\widetilde{\boldsymbol{\Xi}}$ using (34)
  Update $\boldsymbol{\xi}_t$ and $\boldsymbol{\Xi}_t$ via

$$\boldsymbol{\xi}_t = (1-\rho)\boldsymbol{\xi}_{t-1} + \rho\widetilde{\boldsymbol{\xi}}_t$$
$$\boldsymbol{\Xi}_t = (1-\rho)\boldsymbol{\Xi}_{t-1} + \rho\widetilde{\boldsymbol{\Xi}}_t$$

  M step:
  Update $\boldsymbol{\mu}_t$ via (36)
  Update $\mathbf{w}_t$ via (41)
  Let $\mathbf{L}_t = \mathcal{T}\mathbf{w}_t$
end for

updating $\mathbf{L}_t$, we first let $\mathbf{w}_{t-1} = \mathcal{T}^*\mathbf{L}_{t-1}$ and rewrite (37) into a vector form, i.e.,

$$\begin{aligned}\varphi_t(\mathbf{w}) &= \operatorname{tr}\left(\mathcal{T}\mathbf{w}\mathbf{Q}_t\right) - \alpha\mathbf{1}^\top \ln(\mathbf{S}\mathbf{w}) + 2\beta\|\mathbf{w}\|_2^2 + \operatorname{Ind}(\mathbf{w}) \\ &:= \phi_t(\mathbf{w}) + \operatorname{Ind}(\mathbf{w}). \end{aligned} \tag{38}$$

According to the IP-OGD algorithm, the following one-step update is made

$$\mathbf{w}_t = \operatorname{prox}_{\operatorname{Ind}}^{\eta}\left(\mathbf{w}_{t-1} - \eta\nabla\phi_t(\mathbf{w}_{t-1})\right), \tag{39}$$

where $\nabla\phi_t(\mathbf{w}) = \mathcal{T}^*\mathbf{Q}_t - \alpha\mathbf{S}^\top(\mathbf{S}\mathbf{w} + \varepsilon\mathbf{1})^{-1} + 4\beta\mathbf{w}$, ε is a small enough constant, and η is the stepsize. Furthermore,

$$\operatorname{prox}_{\operatorname{Ind}}^{\eta}(\mathbf{v}) = \underset{\mathbf{w}}{\operatorname{argmin}}\left(\operatorname{Ind}(\mathbf{w}) + \frac{1}{2\eta}\|\mathbf{w}-\mathbf{v}\|_2^2\right) \tag{40}$$

is the proximal operator of $\operatorname{Ind}(\cdot)$. Bringing (40) back to (39), we have

$$\mathbf{w}_t = \max\left(\mathbf{w}_{t-1} - \eta\nabla\phi(\mathbf{w}_{t-1}), 0\right), \tag{41}$$

where $\max(\cdot)$ is an element-wise operator. Finally, we let $\mathbf{L}_t = \mathcal{T}\mathbf{w}_t$. The proposed online algorithm is named as **O**nline **EM** algorithm for **G**raph **L**earning from **I**ncomplete **D**ata (OEMGLID), and the complete procedure is placed in Algorithm 2.

4.2. Dynamic regret analysis

Dynamic regret is a common metric for evaluating online algorithms [45,46]. For our problem, let $\boldsymbol{\Phi}_t(\mathbf{L}) = \operatorname{tr}\left(\boldsymbol{\Sigma}^{-1}\mathbf{Q}_t\right) + r(\mathbf{L})$, and the dynamic regret is defined as

$$Reg_d(T) := \sum_{t=1}^{T}\boldsymbol{\Phi}_t(\mathbf{L}_t) - \sum_{t=1}^{T}\boldsymbol{\Phi}_t(\mathbf{L}_t^*), \tag{42}$$

where $\mathbf{L}_t^* = \min_{\mathbf{L}\in\mathcal{L}}\boldsymbol{\Phi}_t(\mathbf{L})$, which represents the optimal action taken by a clairvoyant that knows $\boldsymbol{\Phi}_t(\mathbf{L})$ in advance. By definition, dynamic regret quantifies the cumulative loss incurred by an online algorithm relative to the instantaneous optimal value. An OGL algorithm is admissible only if it yields a sublinear regret since online algorithms with sublinear regret are asymptotically equivalent to batch algorithms [21]. Note that the calculation of (42) depends on the choice of $g(\mathbf{L})$ and $r(\mathbf{L})$. In the following, for better illustration, we will take the smoothness-based GL model (24) as an example. In this case, we can rephrase the dynamic regret in terms of $\mathbf{w}$

$$Reg_d(T) = \sum_{t=1}^{T}\phi_t(\mathbf{w}_t) - \sum_{t=1}^{T}\phi_t(\mathbf{w}_t^*). \tag{43}$$

Next, we derive the dynamic regret of our online algorithm based on the framework of [44]. Before proceeding with the detailed analysis, we make some basic assumptions.

Assumption 3. The estimated sufficient statistics from incomplete data are bounded, i.e., $\|\widetilde{\boldsymbol{\xi}}_t\|_2 \leq B_x$, and $\|\widetilde{\boldsymbol{\Xi}}_t\|_2 \leq B_m, \forall t$.

Assumption 4. The updated graphs are bounded, i.e., $\|\mathbf{w}_t\|_2 \leq B_w, \forall t$.

As stated in [28], Assumption 3 is satisfied when the noise and number of missing values are limited such that the estimated sufficient statistics are bounded by the given constants. In Assumption 4, we place a limit on the magnitude of the updated graph because the graph only makes sense when the edge weights are limited. In practice, we can define a set $\mathcal{W} := \{\mathbf{w} : \mathbf{w} \geq 0, \|\mathbf{w}\|_2 \leq B_w\}$ and rescale the updated graph into this set.

Next, it is observed that $\mathbf{Q}_t$ is estimated from incomplete data and may be inexact. Thus, the gradient $\nabla\phi_t(\mathbf{w})$ is also inexact. Following [44], we define the error in the gradient as

$$\mathbf{e}_t = \nabla\phi_t(\mathbf{w}_t) - \nabla\phi_t^*(\mathbf{w}_t), \tag{44}$$

where $\nabla\phi^*(\mathbf{w})$ is the gradient based on the real complete data. Correspondingly, we define the cumulative gradient error as

$$E(T) = \sum_{t=1}^{T}\|\mathbf{e}_t\|_2, \tag{45}$$

and the path length as

$$P(T) = \sum_{t=2}^{T}\|\mathbf{w}_t^* - \mathbf{w}_{t-1}^*\|_2. \tag{46}$$

Intuitively, the path length measures the cumulative variation of the optimal graph, reflecting the variability of the dynamic environments. Next, we present the following three lemmas based on Assumptions 3–4.

Lemma 1. *The function ϕ_t is L_ϕ-smooth on $\mathcal{W}$, i.e.,*

$$\|\nabla\phi_t(\mathbf{w}) - \nabla\phi_t(\mathbf{w}')\|_2 \leq L_\phi\|\mathbf{w}-\mathbf{w}'\|_2,\ \mathbf{w},\mathbf{w}' \in \mathcal{W}, \tag{47}$$

where $L_\phi = 4\beta + \frac{2\alpha(d-1)}{\varepsilon^2}$.

Lemma 2. *The function ϕ_t is C_ϕ-convex, i.e.,*

$$\left\langle\nabla\phi_t(\mathbf{w}) - \nabla\phi_t(\mathbf{w}'), \mathbf{w}-\mathbf{w}'\right\rangle \geq C_\phi\|\mathbf{w}-\mathbf{w}'\|_2,\ \mathbf{w},\mathbf{w}' \in \mathcal{W}, \tag{48}$$

where $C_\phi = 4\beta$.

Lemma 3. *Under Assumptions 3–4, the gradient $\nabla\phi_t(\mathbf{w}_t)$ is bounded as*

$$\|\nabla\phi_t(\mathbf{w}_t)\|_2 \leq B_\phi, \tag{49}$$

where $B_\phi = 2\sqrt{d-1}(B_x + B_m^2) + 4\beta B_w + \alpha\sqrt{2d(d-1)}/\varepsilon$.

Proof. The proofs of the three lemmas are placed in Appendix B.

Lemmas 1–3 describe the Lipschitz smoothness, strong convexity, and bounded gradient of ϕ_t, which are essential to derive dynamic regret of our online algorithm. Based on these lemmas, we present a bound on the dynamic regret of our online algorithm.

Theorem 1. *Under Assumptions 3–4, let $0 < \eta \leq 1/L_\phi$, then the dynamic regret of our online algorithm for learning graphs from smooth and*

incomplete data is bounded as

$$Reg_d(T) \le \frac{B_\phi}{1-\zeta}\left(\|\mathbf{w}_0 - \mathbf{w}_0^*\|_2 + P(T) + \eta E(T)\right), \tag{50}$$

where $\zeta = 1 - \eta C_\phi$.

Proof. The proof is inspired from [44] but with some key modifications due to the specifics of our problem. According to Theorem 1 in [44], if $0 < \eta \le 1/L_\phi$, we have

$$\begin{aligned}
&\sum_{t=1}^{T} \phi_t(\mathbf{w}_t) - \phi_t(\mathbf{w}_t^*) \le \sum_{t=1}^{T} \left\langle \nabla\phi_t(\mathbf{w}_t), \mathbf{w}_t - \mathbf{w}_t^* \right\rangle \\
&\le \sum_{t=1}^{T} \|\nabla\phi_t(\mathbf{w}_t)\|_2 \|\mathbf{w}_t - \mathbf{w}_t^*\|_2 \le \sum_{t=1}^{T} B_\phi \|\mathbf{w}_t - \mathbf{w}_t^*\|_2 \\
&\le \frac{B_\phi}{1-\zeta}\left(\|\mathbf{w}_0 - \mathbf{w}_0^*\|_2 + P(T) + \eta E(T)\right),
\end{aligned} \tag{51}$$

where $\zeta = 1 - \eta C_\phi$. The first inequality holds due to the first-order convexity condition, and the third inequality holds due to Lemma 3. The last inequality holds due to Lemma 2 in [44]. However, to employ this lemma, we need to verify that our model satisfies the corresponding assumptions. The Lipschitz continuity and strong convexity of ϕ_t have been verified in Lemma 1 and Lemma 2, respectively. Furthermore, the non-expansiveness of $\mathrm{prox}_{\mathrm{Ind}}(\cdot)$, i.e., $\|\mathrm{prox}_{\mathrm{Ind}}(\mathbf{w}) - \mathrm{prox}_{\mathrm{Ind}}(\mathbf{w}')\|_2 \le \|\mathbf{w} - \mathbf{w}'\|_2$, guarantees the feasibility of using Lemma 2 in [44]. Finally, we complete the proof.

Theorem 1 illustrates that the dynamic regret bound of our online algorithm depends on $P(T)$ and $E(T)$. Specifically, $P(T)$ measures the variability of the underlying graphs, and $E(T)$ measures the estimation uncertainty caused by the missing values. Intuitively, if the underlying graph changes rapidly and the missing ratio is high, large $P(T)$ and $E(T)$ will be incurred. In this case, as indicated in (51), our algorithm obtains a higher dynamic regret bound. Moreover, if $P(T)$ and $E(T)$ are sublinear with t, the dynamic regret bound will also be sublinear.

We should mention that Theorem 1 describes the dynamic regrets of the learned graphs learned by our online algorithm based on reconstructed signals. Another challenge is to determine under which conditions our algorithm can identify parameters and missing signals. The identifiability conditions may guide us to obtain lower $E(T)$, which in turn leads to lower $Reg_d(T)$. However, this is out of the scope of this paper and will be left to future works. Moreover, Theorem 1 presents the regret bound on the smoothness-based GL model. The bounds of other models are also left for future work.

5. Experiments

We will test the proposed method using synthetic and real data. First, some experimental setups are presented.

5.1. Experimental setups

(1) Graph generation: We generate Gaussian radial basis function graphs via the method in [9]. Specifically, we generate 20 vertices whose coordinates are randomly located in a unit square. Edge weights are calculated using $\exp(-\mathrm{dist}(i,j)^2/2\sigma_r^2)$, where $\mathrm{dist}(i,j)$ is the distance between vertices i and j, and $\sigma_r = 0.5$ is a kernel width parameter. We remove the edges whose weights are smaller than 0.7.

(2) Signal generation: For the graph with the Laplacian matrix $\mathbf{L}$, we generate T smooth signals from the following distribution [9]

$$\mathbf{x}_t \sim \mathcal{N}\left(\mathbf{0}, \mathbf{L}^\dagger\right), \quad t = 1, \ldots, T. \tag{52}$$

Then, we produce incomplete data via the following missing mechanisms.

MCAR: The missing values depend neither on the observed values nor on missing values. Thus, we randomly mask data with a missing ratio δ to form $\mathbf{X}_o$ and $\mathbf{X}_m$.

MAR: The missing values depend on the observed values. Thus, for each data vector $\mathbf{x}_t$, $\mathbf{x}_t[2 * i]$ is missing when $\mathbf{x}_t[2 * (i-1) + 1] < \delta_m$, where δ_m is a threshold, $i = 1, \ldots, \lfloor d/2 \rfloor$ and $\lfloor \cdot \rfloor$ means rounding down.

MNAR: The missing values depend on the missing values. For each data vector $\mathbf{x}_t$, $\mathbf{x}_t[2 * i]$ is missing when $\mathbf{x}_t[2 * i] < \delta_m$, where $i = 1, \ldots, \lfloor d/2 \rfloor$.

Except for the experiments that test missing mechanisms, all other experiments utilize the MCAR to generate incomplete data.

(3) Evaluation metric: Three metrics are employed in this study. The first is the F1-score (FS), which is a metric for classification tasks since determining whether two vertices are connected can be viewed as a binary classification problem. As shown in (53), TP is the true positive rate, FN is the false negative rate, and FP is the false positive rate. The second metric is the relative error of graph edge weights (ReG), where $\widehat{\mathbf{W}}$ is the learned adjacency matrix, and $\mathbf{W}^*$ is the ground truth. The last metric is the relative error of recovered signals (ReS), where $\mathbf{X}^*$ is the real complete signals, and $\widehat{\mathbf{X}} = [\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_T]$ is the recovered signals.

$$\mathrm{FS} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FN} + \mathrm{FP}}, \mathrm{ReG} = \frac{\|\widehat{\mathbf{W}} - \mathbf{W}^*\|_\mathrm{F}}{\|\mathbf{W}^*\|_\mathrm{F}}, \ \mathrm{ReS} = \frac{\|\widehat{\mathbf{X}} - \mathbf{X}^*\|_\mathrm{F}}{\|\mathbf{X}^*\|_\mathrm{F}}. \tag{53}$$

(4) Baselines: Six baselines are leveraged in the batch case, where LogImp and SigrepImp represent the methods of [9,10] with missing data **imp**uted by the means of the observed data. Moreover, GLEC is the method in [12], but only learns a single graph, and JISG is the joint method for learning graphs based on the SEM model in [13]. JGLSR is the methods that jointly learn graphs based on smoothness assumption and recovers signals via variation minimization [14,38]. ROGL [15] further considers sparse outliers in data. Finally, VBGN [16] is a joint model based on the variational Bayes framework. We consider the mixture of Gaussian (MoG) noise model in this work. For the online case, we use JSTISO [28] as a baseline, which is based on the SVARM model. We also utilize the OGL algorithm in [43]. The missing data is imputed with the mean value of each observed data, which is termed LogImpOnline.

(5) Determination of parameters: As mentioned before, our model requires very few parameters. We only need to determine the parameters in $r(\mathbf{L})$. For the smoothness-based GL model (24), the parameters have a scaling effect [10]. Therefore, α is fixed as 1, and β is selected as the value achieving the best FS. For the online algorithm, $\rho = 0.01$ and $\eta = 0.005$. All parameters of baselines are selected as those achieving the best FS values.

5.2. Synthetic data

(1) The batch case: We first let $T = 100$ and vary δ from 0.1 to 0.5 to test the effect of missing ratios. As illustrated in Fig. 1, FS decreases and ReG increases with the growing δ, which indicates that the graph topology inference performance degrades as δ increases. This is because it is more difficult to recover the complete data with a higher missing ratio, which in turn affects GL performance. However, our method outperforms all baselines for both FS and ReG, implying its superiority in learning graphs from incomplete data. GLEC, LogImp, and SigRepImp fail to obtain satisfactory performance since they separately impute graph signals and infer graph topology, ignoring the underlying connections between the two tasks. Besides, JISG is inferior to ours because it is based on the SEM model and does not comply with the smoothness assumption. VBGN also uses a Bayesian framework like ours to jointly learn the graph and recover the missing signals. However, it obtains inferior performance since it assume a non-Gaussian noise model which may not match our experimental setups. Although JGLSR and ROGL are joint models, our model outperforms them since it can better estimate the missing signals. To verify this, we provide the performance of the signal recovery task in Fig. 1(c). GLEC is omitted since it cannot recover graph signals. Furthermore, we merge LogImp and SigRepImp into Imp because they all use the mean values of the observed data to impute missing data. We can conclude that

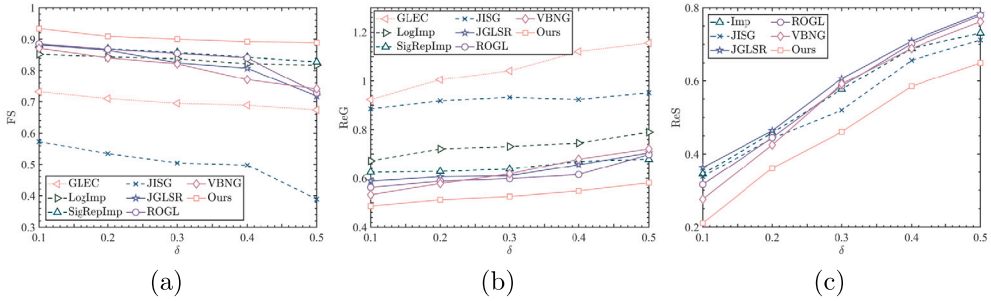

Fig. 1. The results under fixed data sizes and varying missing ratios.

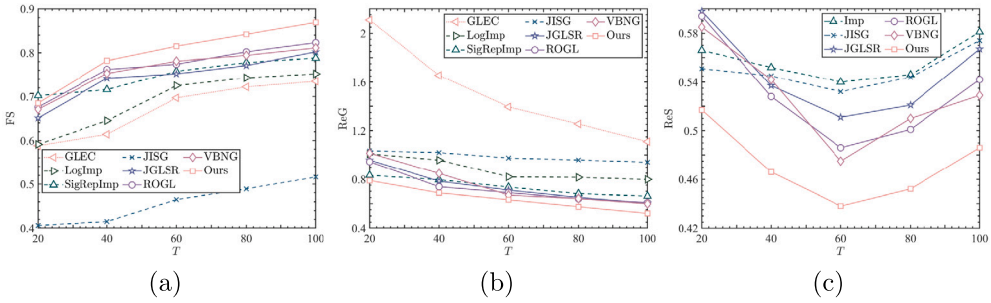

Fig. 2. The results under fixed missing ratios and varying data sizes.

our method achieves the best graph signal recovery performance under different δ, thereby improving the performance of learning graphs.

Next, we fix $\delta = 0.3$ and vary T from 20 to 100 to test the effect of data sizes. As depicted in Fig. 2, all methods achieve improved graph topology inference performance as T increases. JISG obtains the worst FS, while the ReG of GLEC is sensitive to the data sizes. Among all these methods, our method yields the best GL performance, especially for large data sizes. For the graph signal imputation task, ReS first decreases because increasing T results in better graphs, which in turn improves signal inpainting. However, as T continues to increase, ReS starts to increase since the definition of ReS implies that it may also be affected by T. However, our method outperforms other baselines on signal recovery tasks for all T.

We also test the effect of different missing mechanisms. We fix $\delta = 0.3$, $\delta_m = 1$, and $T = 100$. As shown in Fig. 4, the performance of our method in the MNAR mechanism is inferior to that in the other two mechanisms since it is built on the MCAR and MAR mechanisms, as Assumption 2 states. However, compared with other methods, it still achieves the best performance for all three mechanisms.

Finally, we compare the runtime of our algorithm with that of two joint methods, JISG and JGLSR. We vary d from 10 to 1000 and fix $\delta = 0.3, T = 1000$. Besides, we let $K = 100$ and if the relative error between two iterations, $\|\mathbf{W}^{(k+1)} - \mathbf{W}^{(k)}\|_2 / \|\mathbf{W}^{(k)}\|_F$, is smaller than 10^{-4}, the algorithms stop. The algorithms are implemented by MATLAB and run on an Intel(R) CPU with 3.80 GHz clock speed and 16 GB of RAM. The results are shown in Fig. 3. It is observed that the running time of our algorithm is comparable to existing works, which is consistent with the analysis of computational complexity.

(2) The online case: In this case, we test the online algorithm for learning graphs from incomplete data in the presence of topology switching. Specifically, we let $T = 2000$ and change the graph topology abruptly at $t = 1000$. At each time slot, we sequentially receive a data point with missing values. We fix $\delta = 0.3$, and the results are displayed in Fig. 5. It is observed that our method can better track the graphs to lower ReG. When the graph topology switches, it can quickly track the topological changes despite initially suffering from larger ReG. Moreover, JSTISO fails to achieve satisfactory performance since it does not comply with the smoothness assumption. As t increases, the ReG

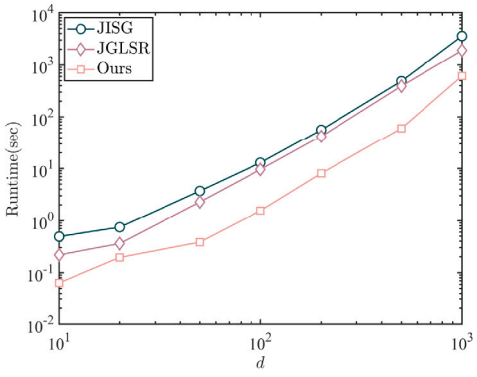

Fig. 3. Comparison of runtime between three joint methods.

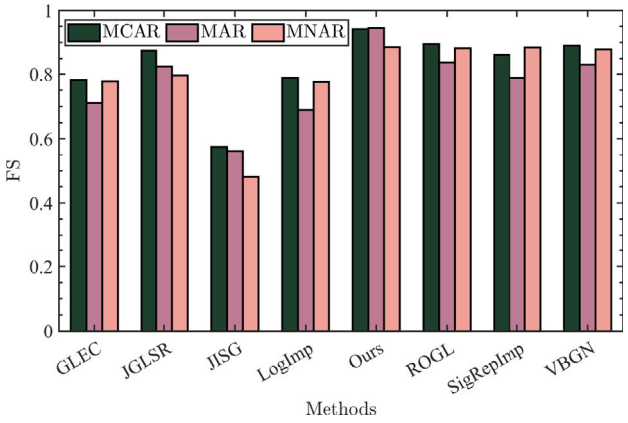

Fig. 4. The topology inference results under the three missing mechanisms.

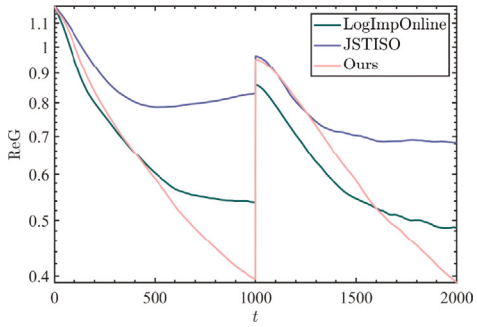

Fig. 5. The ReG results for online learning graphs from incomplete data.

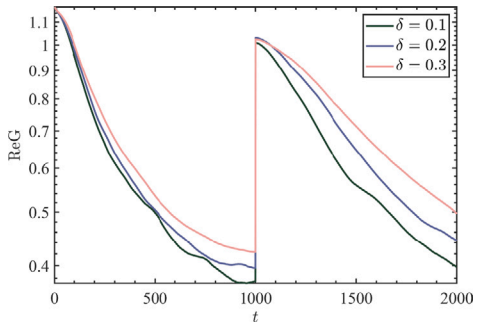

Fig. 6. The ReG results of our online method for different missing ratios.

of LogImpOnline cannot be further reduced because simply imputing missing values by mean replacement cannot recover missing signal values accurately.

In Fig. 6, we test the effect of different δ on the tracking performance. It is observed that increasing δ reduces the tracking ability since large δ may lead to large gradient error, as suggested in Theorem 1. However, for all three missing ratios, our algorithm can effectively track the dynamic graph.

Finally, we compare the proposed online algorithm with the batch algorithm on static graphs. We fix $\delta = 0.3$ and vary T from 50 to 1000. For the online algorithm, we use the results of the final round to obtain the best performance. As illustrated in 7, when T is small, the batch algorithm is far superior to the online algorithm. As [28] states, an online formulation can produce a sequence of results with minimum delay at the price of lower accuracy. The reason is that in each time slot of online algorithms, only a few updates are performed based on one newly arrived data due to delay constraints. On the contrary, the batch algorithm can utilize all data to update graphs until convergence. However, as T increases, the online algorithm will approach the performance of the batch algorithm since the online one obtains enough rounds to update. Furthermore, the online algorithm can track dynamic graphs that batch algorithms cannot handle.

5.3. Real data

In this work, we apply our work to real-world datasets from two application domains, i.e., medical and social networks. We can observe from Assumptions 1–2 that our method is particularly suitable for the signals that encode structured information, such as the smooth graph signals in the following two datasets. Furthermore, the missing values should be (completely) at random.

(1) The batch case: In this experiment, we learn brain functional connectivity graphs to explore the impact of autism on brain functional connectivity. We use the blood-oxygenation-level-dependent (BOLD) time series extracted from fMRI data as graph signals. The basic assumption is that if two functional areas of a brain are strongly connected, their BOLD data should be similar. The corresponding two nodes in the learned graph are hence connected. Furthermore, autism may affect brain functional connectivity. If we can learn about the differences in brain functional connectivity graphs between autistic and non-autistic people, it may help us better understand the pathogenesis of autism. The dataset[2] contains 539 autistic individuals and 573 typical controls, from which we select an autistic and a non-autistic subject. Each subject contains 176 signals (the length of the BOLD time series is 176). Besides, we select 34 functional regions of interest from 90 standard regions of the Anatomical Automatic Labeling (AAL) template, indicating that $\mathbf{X} \in \mathbb{R}^{34\times 176}$. We randomly mask 30% data by following the MCAR mechanism. As a competing method, we learn the graphs using all complete data via [10]. As depicted in Fig. 8, there are some noticeable topological changes between the graphs of autistic and non-autistic individuals. These edges reflect changes in the functional connectivity caused by autism, which may aid in diagnosing autism. Similar to the method with complete data, our algorithm successfully captures topological changes caused by autism despite missing values, demonstrating the effectiveness of our method.

(2) The online case: Finally, we apply our method to social networks,[3] the roll call data of senators in the USA from the 103rd Congress to the 111th Congress(1993 to 2010), to online track graphs from incomplete data. We choose 30 senators who have been in Parliament during this time, and 15 of them are Republicans while the remaining are Democrats. From 1993 to 2011, a total of 6233 proposals were made, and we take the "yea" or "nay" votes of each senator on these proposals as signals. For each proposal, there may be missing values such as abstention votes. We use 1 and -1 to represent "yea" and "nay", respectively. The other vote results are taken as missing values. We use this dataset to understand the relationships between these senators, based on the assumption that senators with close social connections should have similar political positions. Thus, we can use the roll call data of these senators to infer the underlying social networks between them. Fig. 9 depicts the relationships between the selected senators of different periods. Two clusters are shown in Fig. 9(a), and the senators who belong to the same party have closer relationships while those from different parties have fewer connections. This trend makes sense because senators from the same party tend to hold similar political positions. In April 2001, which is shown in Fig. 9(b), the connections between the two parties are cut off, indicating that the political differences between the two parties were even greater at this time. In 2010, the connections between the two parties were re-established. Note that although Senators 11 and 15 are Republicans, they seem to have strong connections to Republican senators. We find that Senator 15 has changed from a Republican to a Democrat during this period. That explains why he acted like a Democrat. Furthermore, the partner senator (senator of the same state) of Senator 11 changed in the 110th Congress. The former partner was a Republican while the current one is a Democrat. This may affect the vote results made by Senator 11. In summary, our method can track the evolution of the social network in the presence of missing values.

6. Conclusion

In this paper, we presented a framework to learn graphs from incomplete data. We first formulated the problem as maximizing the posterior distribution from a Bayesian perspective. Then, we proposed an EM algorithm to solve the induced problem. Furthermore, we extended the algorithm to an online version to handle the delay-sensitive scenarios of sequential data. The dynamic regret of the online algorithm was analyzed, demonstrating that the regret becomes sublinear when

[2] http://preprocessed-connectomes-project.org/abide/
[3] http://www.voteview.com/dwnl.htm

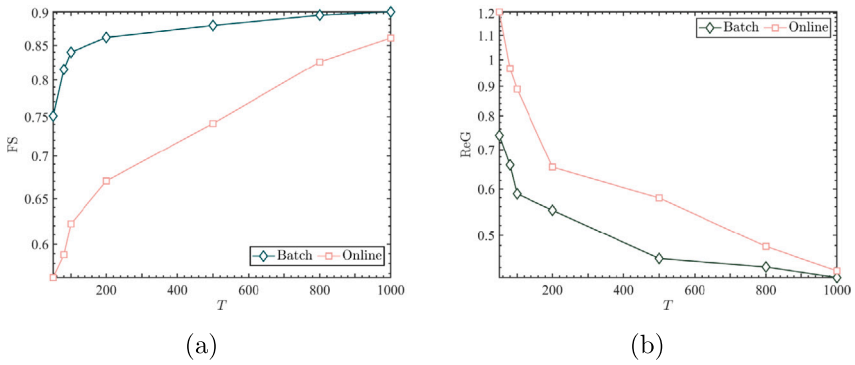

Fig. 7. Comparison between batch and online algorithm.

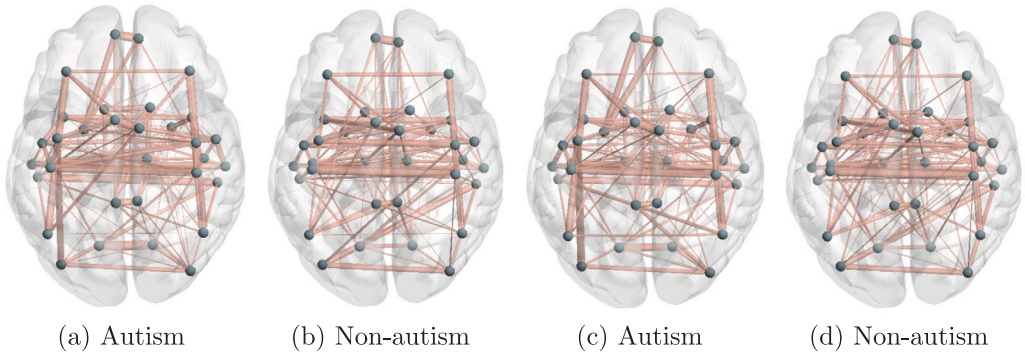

Fig. 8. The learned brain functional connection networks: (a)–(b) the networks learned from complete data; (c)–(d) the networks learned from incomplete data via our method.

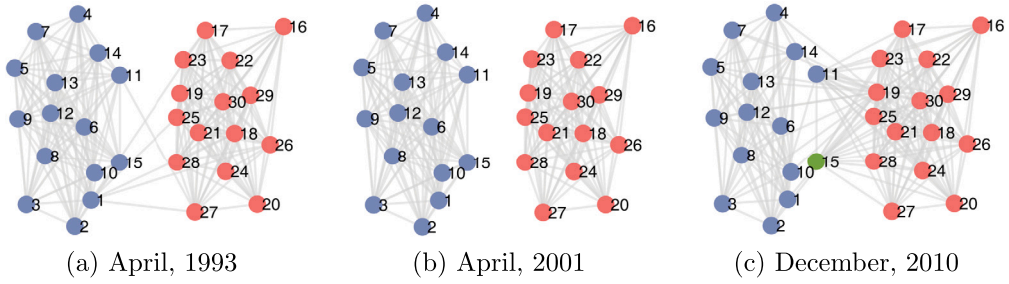

Fig. 9. The social networks between senators at different periods. Red nodes represent Republicans while blue nodes represent Democrats. The only green node in the networks in December 2010 represents the senator who changed from a Republican to a Democrat.

the environmental variation and gradient estimation errors caused by missing data are sublinear. Extensive experiments corroborated that our approach can efficiently learn graphs from incomplete data in batch and online scenarios. Our method is based on the assumption that missing locations are known in advance, which can be further extended to the case where missing locations are unknown, i.e., the blind missing samples [47]. Moreover, future research can also include extending our framework to other scenarios like the graph Laplacian mixture model.

CRediT authorship contribution statement

Xiang Zhang: Writing – original draft, Visualization, Validation, Software, Methodology, Investigation, Formal analysis, Conceptualization. **Qiao Wang:** Writing – review & editing, Validation, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgements

We sincerely thank the reviewers of this paper for their extremely helpful comments. This work is supported in part by the Project Four of NNSF of China Major Program under Grant No. 52394244 and Jiangsu Provincial key research and development program under Grant No. BE2023799.

Appendix A. Definition of operator $\mathcal{T}$

Given a vector $\mathbf{a} \geq 0 \in \mathbb{R}^{d(d-1)/2}$, linear operator $\mathcal{T}$ is used to convert $\mathbf{a}$ to a Laplacian matrix, i.e., $\mathcal{T}\mathbf{a} \in \mathcal{L}$, which is defined as [37]:

$$(\mathcal{T}\mathbf{a})[ij] = \begin{cases} -\mathbf{a}[i + b_j] & i > j, \\ (\mathcal{T}\mathbf{a})[ji] & i < j, \\ -\sum_{i \neq j} (\mathcal{T}\mathbf{a})[ij] & i = j, \end{cases} \tag{A.1}$$

where $b_j = -j + \frac{j-1}{2}(2d-j)$. Accordingly, the adjoint operator $\mathcal{T}^*$ of $\mathcal{T}$ is to transform a matrix $\mathbf{A} \in \mathbb{R}^{d\times d}$ to a vector, which is defined as

$$(\mathcal{T}^*\mathbf{A})[k] = \mathbf{A}[ii] - \mathbf{A}[ij] - \mathbf{A}[ji] + \mathbf{A}[jj], \tag{A.2}$$

where $k = i - j + \frac{j-1}{2}(2d-j)$ with $i > j$. Then, we have Lemma 4.

Lemma 4. *Given a matrix* $\mathbf{A} \in \mathbb{R}^{d\times d}$, $\|\mathcal{T}^*\mathbf{A}\|_2 \le 2\sqrt{d-1}\|\mathbf{A}\|_F$.

Proof. By the definition of $\mathcal{T}^*$, we have

$$\begin{aligned}\|\mathcal{T}^*\mathbf{A}\|_2^2 &= \sum_{k=1}^{d(d-1)/2} (\mathcal{T}^*\mathbf{A})^2[k] = \sum_{k=1}^{d(d-1)/2} (\mathbf{A}[ii] - \mathbf{A}[ij] - \mathbf{A}[ji] + \mathbf{A}[jj])^2 \\ &\le \sum_{k=1}^{d(d-1)/2} 4\left(\mathbf{A}^2[ii] + \mathbf{A}^2[ij] + \mathbf{A}^2[ji] + \mathbf{A}^2[jj]\right) \\ &= \sum_{i\neq j} 4\mathbf{A}^2[ij] + \sum_{i=1}^{d} 4(d-1)\mathbf{A}^2[ii] \\ &\le 4(d-1)\sum_{i,j=1}^{d} \mathbf{A}^2[ij] = 4(d-1)\|\mathbf{A}\|_F^2,\end{aligned} \tag{A.3}$$

where $k = i - j + \frac{j-1}{2}(2d-j), i > j$, and the first inequality holds due to elementary inequality.

Appendix B. Proofs of Lemmas 1–3

Proof of Lemma 1. For any $\mathbf{w}, \mathbf{w}' \in \mathcal{W}$, we have

$$\begin{aligned}&\left\|\nabla\phi_t(\mathbf{w}) - \nabla\phi_t(\mathbf{w}')\right\|_2 \\ =\ &\left\|4\beta(\mathbf{w}-\mathbf{w}') - \alpha\mathbf{S}^\top\left(\frac{1}{\mathbf{S}\mathbf{w}+\varepsilon\mathbf{1}} - \frac{1}{\mathbf{S}\mathbf{w}'+\varepsilon\mathbf{1}}\right)\right\|_2 \\ \le\ & 4\beta\left\|\mathbf{w}-\mathbf{w}'\right\|_2 + \alpha\left\|\mathbf{S}^\top\right\|_2\left\|\frac{1}{\mathbf{S}\mathbf{w}+\varepsilon\mathbf{1}} - \frac{1}{\mathbf{S}\mathbf{w}'+\varepsilon\mathbf{1}}\right\|_2 \\ \le\ & 4\beta\left\|\mathbf{w}-\mathbf{w}'\right\|_2 + \frac{\alpha\|\mathbf{S}\|_2}{\varepsilon^2}\left\|\mathbf{S}\mathbf{w}-\mathbf{S}\mathbf{w}'\right\|_2 \\ \le\ & 4\beta\left\|\mathbf{w}-\mathbf{w}'\right\|_2 + \frac{\alpha\|\mathbf{S}\|_2^2}{\varepsilon^2}\left\|\mathbf{w}-\mathbf{w}'\right\|_2 \\ =\ &\left(4\beta + \frac{2\alpha(d-1)}{\varepsilon^2}\right)\|\mathbf{w}-\mathbf{w}'\|_2 := L_\phi\|\mathbf{w}-\mathbf{w}'\|_2,\end{aligned} \tag{B.1}$$

where the last inequality holds since $\|\mathbf{S}\|_2 = \sqrt{2(d-1)}$ [43].

Proof of Lemma 2. For any $\mathbf{w} \in \mathcal{W}$, we can calculate the Hessian matrix of $\phi_t(\mathbf{w})$ as $\nabla_{\mathbf{ww}}\phi_t(\mathbf{w}) = 4\beta\mathbf{I} + \alpha\mathbf{S}^\top\mathrm{diag}\left((\mathbf{S}\mathbf{w}+\varepsilon\mathbf{1})^{-2}\right)\mathbf{S}$. It is easy to check that $\nabla_{\mathbf{ww}}\phi_t(\mathbf{w}) \succ 4\beta\mathbf{I}$. Hence, $\phi_t(\mathbf{w})$ is 4β–strongly convex.

Proof of Lemma 3. Clearly, we have

$$\begin{aligned}\xi_t &= (1-\rho)\xi_{t-1} + \rho\widetilde{\xi}_t = \rho[(1-\rho)^{t-1}\widetilde{\xi}_1 + (1-\rho)^{t-2}\widetilde{\xi}_2 + \cdots + \widetilde{\xi}_t] \\ &= \rho\sum_{\tau=1}^{t}(1-\rho)^{t-\tau}\widetilde{\xi}_\tau.\end{aligned} \tag{B.2}$$

Also, we have $\Xi_t = \rho\sum_{\tau=1}^{t}(1-\rho)^{t-\tau}\widetilde{\Xi}_\tau$. The norm of ξ_t is bounded by

$$\begin{aligned}\|\xi_t\|_2 &= \left\|\rho\sum_{\tau=1}^{t}(1-\rho)^{t-\tau}\widetilde{\xi}_\tau\right\|_2 \le \rho B_x\left(\sum_{\tau=1}^{t}(1-\rho)^{t-\tau}\right) \\ &= (1-(1-\rho)^t)B_x \le B_x.\end{aligned} \tag{B.3}$$

Similarly, $\|\Xi_t\|_F \le B_m$. It is not difficult to yield

$$\|\mathbf{Q}\|_F \le \|\Xi\|_F + \|\xi\xi^\top\|_F = B_x + B_m^2. \tag{B.4}$$

The gradient $\nabla\phi_t(\mathbf{w}_t)$ can then be bounded as

$$\begin{aligned}\|\nabla\phi_t(\mathbf{w}_t)\|_2 &\le \left\|\mathcal{T}^*\mathbf{Q}_t\right\|_2 + 4\beta\left\|\mathbf{w}_t\right\|_2 + \alpha\left\|\mathbf{S}(\mathbf{S}\mathbf{w}_t+\varepsilon\mathbf{1})^{\cdot(-1)}\right\|_2 \\ &\le 2\sqrt{d-1}\|\mathbf{Q}_t\|_F + 4\beta\left\|\mathbf{w}_t\right\|_2 + \alpha\|\mathbf{S}\|_2\left\|(\mathbf{S}\mathbf{w}_t+\varepsilon\mathbf{1})^{\cdot(-1)}\right\|_2 \\ &\le 2\sqrt{d-1}(B_x + B_m^2) + 4\beta B_w + \alpha\sqrt{2d(d-1)}/\varepsilon \\ &= B_\phi.\end{aligned} \tag{B.5}$$

The second inequality holds due to Lemma 4. The last inequality holds since Assumptions 3 and 4.

References

[1] G. Mateos, S. Segarra, A.G. Marques, A. Ribeiro, Connecting the dots: Identifying network structure via graph signal processing, IEEE Signal Process. Mag. 36 (3) (2019) 16–43.

[2] X. Dong, D. Thanou, M. Rabbat, P. Frossard, Learning graphs from data: A signal representation perspective, IEEE Signal Process. Mag. 36 (3) (2019) 44–63.

[3] U. Von Luxburg, A tutorial on spectral clustering, Stat. Comput. 17 (2007) 395–416.

[4] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, S.Y. Philip, A comprehensive survey on graph neural networks, IEEE Trans. Neural Netw. Learn. Syst. 32 (1) (2020) 4–24.

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

[6] D. Kaplan, Structural Equation Modeling: Foundations and Extensions, vol. 10, SAGE publications, 2008.

[7] G. Chen, D.R. Glen, Z.S. Saad, J.P. Hamilton, M.E. Thomason, I.H. Gotlib, R.W. Cox, Vector autoregression, structural equation modeling, and their synthesis in neuroimaging data analysis, Comput. Biol. Med. 41 (12) (2011) 1142–1155.

[8] A. Ortega, P. Frossard, J. Kovačević, J.M. Moura, P. Vandergheynst, Graph signal processing: Overview, challenges, and applications, Proc. IEEE 106 (5) (2018) 808–828.

[9] X. Dong, D. Thanou, P. Frossard, P. Vandergheynst, Learning Laplacian matrix in smooth graph signal representations, IEEE Trans. Signal Process. 64 (23) (2016) 6160–6173.

[10] V. Kalofolias, How to learn a graph from smooth signals, in: Proc. Int. Conf. Artif. Intell. Stat., AISTATS, PMLR, 2016, pp. 920–929.

[11] X. Pu, T. Cao, X. Zhang, X. Dong, S. Chen, Learning to learn graph topologies, Proc. Adv. Neural Inf. Process. Syst. 34 (2021) 4249–4262.

[12] X. Yang, M. Sheng, Y. Yuan, T.Q. Quek, Network topology inference from heterogeneous incomplete graph signals, IEEE Trans. Signal Process. 69 (2020) 314–327.

[13] V.N. Ioannidis, Y. Shen, G.B. Giannakis, Semi-blind inference of topologies and dynamical processes over dynamic graphs, IEEE Trans. Signal Process. 67 (9) (2019) 2263–2274, http://dx.doi.org/10.1109/TSP.2019.2903025.

[14] P. Berger, G. Hannak, G. Matz, Efficient graph learning from noisy and incomplete data, IEEE Trans. Signal. Inf. Process. Netw. 6 (2020) 105–119.

[15] A. Karaaslanli, S. Aviyente, Graph learning from noisy and incomplete signals on graphs, in: Proc. IEEE Statist. Signal Process. Workshop, IEEE, 2021, pp. 556–560.

[16] R. Torkamani, H. Zayyani, F. Marvasti, Joint topology learning and graph signal recovery using variational Bayes in non-Gaussian noise, IEEE Trans. Circuits Syst. II 69 (3) (2021) 1887–1891.

[17] A. Javaheri, A. Amini, F. Marvasti, D.P. Palomar, Joint signal recovery and graph learning from incomplete time-series, 2023, arXiv preprint arXiv:2312.16940.

[18] A. Javaheri, A. Amini, F. Marvasti, D.P. Palomar, Learning spatio-temporal graphical models from incomplete observations, IEEE Trans. Signal Process. (2024).

[19] E. Kummerfeld, D. Danks, Tracking time-varying graphical structure, Proc. Adv. Neural Inf. Process. Syst. 26 (2013).

[20] S. Yang, H. Xiong, Y. Zhang, Y. Ling, L. Wang, K. Xu, Z. Sun, OGM: Online Gaussian graphical models on the fly, Appl. Intell. 52 (3) (2022) 3103–3117.

[21] B. Zaman, L.M.L. Ramos, D. Romero, B. Beferull-Lozano, Online topology identification from vector autoregressive time series, IEEE Trans. Signal Process. (2020) 210–225.

[22] A. Natali, E. Isufi, M. Coutino, G. Leus, Online graph learning from time-varying structural equation models, in: Proc. IEEE Asilomar Conf. Signals Syst. Comput., IEEE, 2021, pp. 1579–1585.

[23] S.S. Saboksayr, G. Mateos, M. Cetin, Online graph learning under smoothness priors, in: Proc. Eur. Signal Process. Conf., IEEE, 2021, pp. 1820–1824.

[24] A. Natali, E. Isufi, M. Coutino, G. Leus, Learning time-varying graphs from online data, IEEE Open J. Signal Process. 3 (2022) 212–228.

[25] X. Zhang, Q. Wang, Online graph learning in dynamic environments, in: Proc. Eur. Signal Process. Conf., IEEE, 2022, pp. 2151–2155.

[26] S.S. Saboksayr, G. Mateos, Dual-based online learning of dynamic network topologies, 2022, arXiv preprint arXiv:2211.07449.

[27] R. Money, J. Krishnan, B. Beferull-Lozano, Online joint nonlinear topology identification and missing data imputation over dynamic graphs, in: Proc. Eur. Signal Process. Conf., IEEE, 2022, pp. 687–691.

[28] B. Zaman, L.M. Lopez-Ramos, B. Beferull-Lozano, Online joint topology identification and signal estimation from streams with missing data, IEEE Trans. Signal. Inf. Process. Netw. (2023).

[29] R.J. Little, D.B. Rubin, Statistical Analysis with Missing Data, vol. 793, John Wiley & Sons, 2019.
[30] M. Aste, M. Boninsegna, A. Freno, E. Trentin, Techniques for dealing with incomplete data: a tutorial and survey, Pattern Anal. Appl. 18 (2015) 1–29.
[31] O. Cappé, E. Moulines, Online expectation–maximization algorithm for latent data models, J. R. Stat. Soc. B 71 (3) (2009) 593–613.
[32] L. Stanković, M. Daković, E. Sejdić, Introduction to graph signal processing, in: Vertex-Frequency Analysis of Graph Signals, Springer, 2019, pp. 3–108.
[33] H.P. Maretic, P. Frossard, Graph Laplacian mixture model, IEEE Trans. Signal. Inf. Process. Netw. 6 (2020) 261–270.
[34] N. Städler, P. Bühlmann, Missing values: sparse inverse covariance estimation and an extension to sparse regression, Stat. Comput. 22 (2012) 219–235.
[35] G. Marsaglia, Conditional means and covariances of normal variables with singular covariance matrix, J. Amer. Statist. Assoc. 59 (308) (1964) 1203–1204.
[36] S.S. Saboksayr, G. Mateos, Accelerated graph learning from smooth signals, IEEE Signal Process. Lett. 28 (2021) 2192–2196.
[37] S. Kumar, J. Ying, J.V. de Miranda Cardoso, D.P. Palomar, A unified framework for structured graph learning via spectral constraints, J. Mach. Learn. Res. 21 (22) (2020) 1–60.
[38] S. Chen, A. Sandryhaila, J.M. Moura, J. Kovačević, Signal recovery on graphs: Variation minimization, IEEE Trans. Signal Process. 63 (17) (2015) 4609–4624.
[39] S. Chen, A. Sandryhaila, G. Lederman, Z. Wang, J.M. Moura, P. Rizzo, J. Bielak, J.H. Garrett, J. Kovačevic, Signal inpainting on graphs via total variation minimization, in: Proc. IEEE Int. Conf. Acoust., Speech, Signal Process., IEEE, 2014, pp. 8267–8271.
[40] P. Berger, G. Hannak, G. Matz, Graph signal recovery via primal-dual algorithms for total variation minimization, IEEE J. Sel. Topics Signal Process. 11 (6) (2017) 842–855.
[41] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, Y. Gu, Time-varying graph signal reconstruction, IEEE J. Sel. Topics Signal Process. 11 (6) (2017) 870–883.
[42] X. Zhang, Online graph learning in dynamic environments, 2021, arXiv preprint arXiv:2110.05023.
[43] S.S. Saboksayr, G. Mateos, M. Cetin, Online discriminative graph learning from multi-class smooth signals, Signal Process. 186 (2021) 108101.
[44] R. Dixit, A.S. Bedi, R. Tripathi, K. Rajawat, Online learning with inexact proximal online gradient descent algorithms, IEEE Trans. Signal Process. 67 (5) (2019) 1338–1352.
[45] E. Hazan, et al., Introduction to online convex optimization, Found. Trends Mach. Learn. 2 (3–4) (2016) 157–325.
[46] E.C. Hall, R.M. Willett, Online convex optimization in dynamic environments, IEEE J. Sel. Topics Signal Process. 9 (4) (2015) 647–662.
[47] H. Zayyani, M. Korki, F. Marvasti, Bayesian hypothesis testing detector for one bit diffusion LMS with blind missing samples, Signal Process. 146 (2018) 61–65.