Kernel-Based Graph Learning From Smooth Signals: A Functional Viewpoint

Xingyue Pu, Siu Lun Chau, Xiaowen Dong, and Dino Sejdinovic

***Abstract*—The problem of graph learning concerns the construction of an explicit topological structure revealing the relationship between nodes representing data entities, which plays an increasingly important role in the success of many graph-based representations and algorithms in the field of machine learning and graph signal processing. In this paper, we propose a novel graph learning framework that incorporates prior information along node and observation side, and in particular the covariates that help to explain the dependency structures in graph signals. To this end, we consider graph signals as functions in the reproducing kernel Hilbert space associated with a Kronecker product kernel, and integrate functional learning with smoothness-promoting graph learning to learn a graph representing the relationship between nodes. The functional learning increases the robustness of graph learning against missing and incomplete information in the graph signals. In addition, we develop a novel graph-based regularisation method which, when combined with the Kronecker product kernel, enables our model to capture both the dependency explained by the graph and the dependency due to graph signals observed under different but related circumstances, e.g. different points in time. The latter means the graph signals are free from the i.i.d. assumptions required by the classical graph learning models. Experiments on both synthetic and real-world data show that our methods outperform the state-of-the-art models in learning a meaningful graph topology from graph signals, in particular with heavy noise, missing values, and multiple dependency.**

***Index Terms*—Functional viewpoint, graph learning, graph signal processing, kernel methods.**

I. Introduction

MODELLING based on graphs has recently attracted an increasing amount of interest in machine learning and signal processing research. On the one hand, many real-world data are intrinsically graph-structured, e.g. individual preferences in social networks or environmental monitoring data from sensor networks. This makes graph-based methods a natural approach to analysing such structured data. On the other hand, graphs are an effective modelling language for revealing relational structure in complex domains and may assist in a variety of learning tasks. For example, knowledge graphs improve the performance in semantic parsing and question answering [1]. Despite their usefulness, however, a graph is not always readily available or explicitly given. The problem of graph learning therefore concerns the construction of a topological structure among entities from a set of observations on these entities.

Methodologies to learn a graph from the structured data include naïve methods such as k-nearest neighbours (k-NN), and approaches from the literature of probabilistic graphical models (PGMs) and more recently graph signal processing (GSP) and graph neural networks (GNNs). The basic idea of k-NN is to connect a node to k other nodes with the smallest pairwise distances in terms of the observations [2]–[5]. In PGMs, a graph expresses the conditional dependence with edges between random variables represented by nodes [6]. The GSP literature, on the other hand, focuses on algebraic and spectral characteristics of the graph signals [7]–[9], which are defined as observations on a collection of nodes. The GSP-based graph learning methods (see [10], [11] for two recent reviews) further fall into two distinct branches, i.e. those based on the diffusion processes on graphs [12]–[15] and those based on smoothness measures of graph signals [16]–[21]. Very recently, GNNs have attracted a surging interest in the machine learning community which leads to a number of approaches to graph inference [22], [23].

While many of the above methods can effectively learn a meaningful graph from observations, there is a lack of consideration of the prior information, e.g. node-side covariates, which may be available for the task at hand. Those covariates that provide valuable side information should be integrated into the graph learning framework. Taking an example of measuring temperature records in different locations in a country, where nodes represent weather stations, the latitude, longitude and altitude of each station are useful node-side information. One major benefit is to lessen the reliance of the above models on the quality of the observations. Heavily corrupted or even missing records can be predicted by such prior information, which in turn helps improve the efficiency in graph inference.

Besides the node-side prior information, the observation-side dependency is also largely ignored in the literature. One example are temperature records collected at different timestamps, which are clearly related and could largely affect the evaluation of the strength of relation between stations. Another example is that of a recommender system, where the item ratings collected from different individuals are largely affected by the social relationship between them.

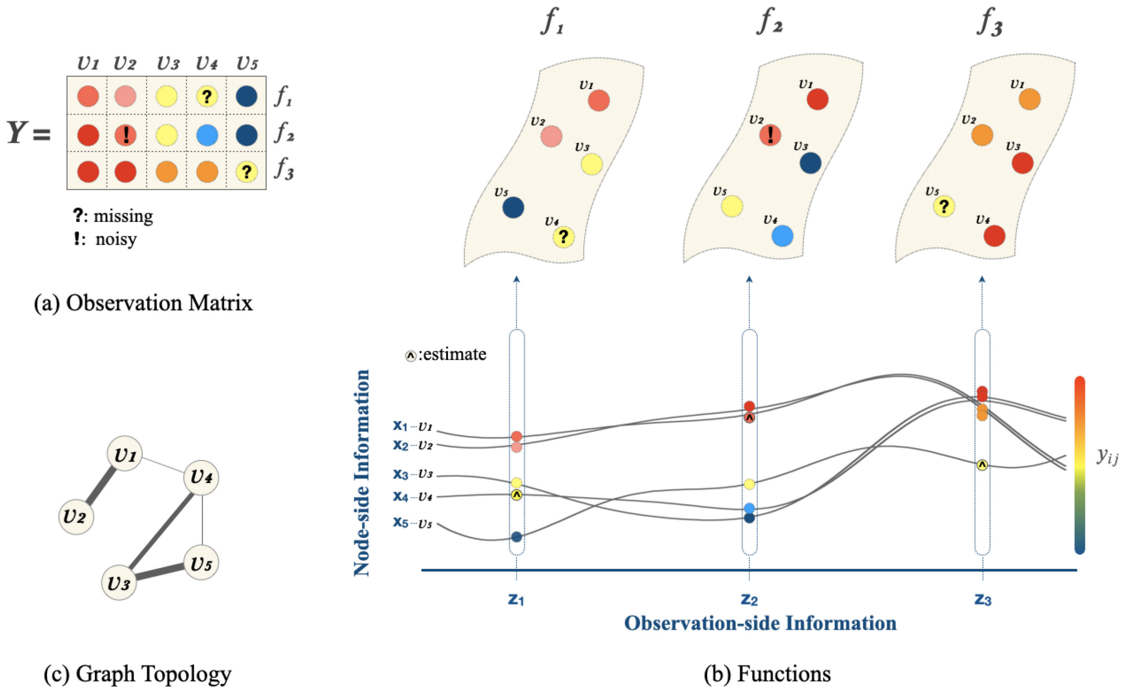
Manuscript received August 23, 2020; revised December 20, 2020; accepted February 11, 2021. Date of publication February 17, 2021; date of current version March 18, 2021. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Tirza Routtenberg. *(Corresponding author: Xingyue Pu.)*

Xingyue Pu and Xiaowen Dong are with the Oxford-Man Institute, and the Department of Engineering Science, University of Oxford, Oxford OX2 6ED, U.K. (e-mail: xpu@robots.ox.ac.uk; xdong@robots.ox.ac.uk).

Siu Lun Chau and Dino Sejdinovic are with the Department of Statistics, University of Oxford, Oxford OX1 3LB, U.K. (e-mail: siu.chau@stats.ox.ac.uk; dino.sejdinovic@stats.ox.ac.uk).

This article has supplementary downloadable material available at https://doi.org/10.1109/TSIPN.2021.3059995, provided by the authors.

Digital Object Identifier 10.1109/TSIPN.2021.3059995



(a) Observation Matrix

(c) Graph Topology

(b) Functions

Fig. 1. A functional viewpoint of graph learning: (a) The observation matrix with missing and noisy entries; (b) Each row of the observation matrix is modelled as samples obtained at a collection of fixed locations (considered as nodes in a graph) from an underlying function f (top); Values at each node are determined by the underlying function, as well as node-side information $\mathbf{x}$ and observation-side information $\mathbf{z}$ (bottom); (c) The learned graph topology.

To tackle the above issues, we revisit the graph signal observations from a functional viewpoint and propose a framework for learning undirected graphs by considering additional covariates on both the node- and observation-side. This allows us to capture dependency structure beyond the graph signals which leads to more effective and graph and signal recovery. More specifically, as shown in Fig. 1, the ij-th entry of the graph-structured data matrix $\mathbf{Y} \in \mathbb{R}^{n\times m}$, which contains n graph signals collected on m nodes, can be viewed as some potentially noisy or missing observation of $f_i(j)$, i.e. the i-th function evaluated at j-th node. To model the node-side information, we introduce a covariate $\mathbf{x} \in \mathcal{X}$ that can explain the variations in a graph signal, e.g. a vector that contains the latitude, longitude and altitude of stations in the aforementioned temperature example. To model the observation-side information, we also introduce a generic covariate $\mathbf{z} \in \mathcal{Z}$. For example, $\mathbf{z}$ could be the timestamp at which the temperature record is collected. Observation-side dependency hence arises due to f_i depending on $\mathbf{z}_i$. Combining the two, the function underlying the graph signals takes the form of $f_i(j) = f(\mathbf{z}_i, \mathbf{x}_j)$.

Formally, we define a function $f : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$ in a reproducing kernel Hilbert space (RKHS) with a product kernel $\kappa_\otimes = \kappa_\mathcal{Z} \otimes \kappa_\mathcal{X}$ on $\mathcal{Z} \times \mathcal{X}$, where $\otimes$ denotes the Kronecker product. The function draws a connection between graph signals and both node- and observation-side covariates. At the same time, by modifying the classical Laplacian quadratic form that is widely adopted in the literature, we propose a novel smoothness measure that takes into account the observation-side dependency introduced by $\mathbf{z} \in \mathcal{Z}$, which has a nice interpretation associated with a Kronecker product graph. Our key contribution is the Kernel Graph Learning (**KGL**) framework, which allows us to learn the graph Laplacian matrix $\mathbf{L}$ by jointly inferring the function f and optimising the smoothness over $\mathbf{L}$ of the graph signals modelled and refined by f with node- and observation-side prior information.

In addition, we provide several extensions of **KGL** for the scenario of a partially observed $\mathbf{Y}$ with known missing value positions, and that of observations without either node-side or observation-side information. The learning problem is effectively solved via a block coordinate descent algorithm, which has a theoretical guarantee of convergence. We show that **KGL** can effectively recover the groundtruth graph from the two-side dependent data and outperform the state-of-the-art smoothness-based graph learning methods in both synthetic and real-world experiments.

In summary, the main contributions of our work are as follows:

- A novel graph-based regularisation based on a smoothness measure of graph signals with observation-side dependency;
- A graph learning framework that integrates node- and observation-side covariates from a functional viewpoint;
- An efficient method for denoising and imputing missing values in the observed graph signals as a byproduct of the graph learning framework.

II. Related Work

In this section, we survey the classical methods of learning a graph from a number of different perspectives. From each perspective, we highlight the most related work that considers one or more aspects of the 1) node-side information, 2) observation-side dependency, and 3) noisy and missing data.

A. k-Nearest Neighbours Methods

The k-nearest neighbours (k-NN) connects a node to k other nodes with the smallest pairwise distances in terms of the observations. It is flexible with different choices of distance metrics and yet heuristic since the neighbourhood search is based on pairwise comparison of observations on nodes. The majority of the k-NN variants focuses on fast approximate search algorithms (ANN) [2]–[5] and recent variants apply deep reinforcement learning to explicitly maximise search efficiency [24], [25]. By comparison, the model-based methods, e.g. PGMs and GSP, directly integrate global properties of the observations into learning objectives.

B. Probabilistic Graphical Models

In the field of PGMs, the inverse covariance matrix Θ is often regarded as an undirected graph that parameterises the joint distribution of random variables representing nodes. There is a rich literature on effective algorithms to estimate a sparse Θ from Gaussian-distributed data by solving an ℓ_1-regularised log-likelihood maximisation [26]–[30], including the widely used graphical Lasso [31] and G-ISTA [32]. The recent state-of-the-art algorithm BigQUIC [33] scales to millions of nodes with performance guarantees. Besides computational improvements, models based on attractive Gaussian Markov Random Fields (GMRFs) [34]–[37] further restrict the off-diagonal entries of Θ to be non-positive, which is equivalent to learning the Laplacian matrix of the corresponding graph with non-negative edge weights. The most related extensions of the graphical Lasso were proposed in [38], [39], which simultaneously learn two dependency structures in the matrix-variate Gaussian data. While their work focuses on estimating covariance matrices, our work focuses on recovering a graph topology from data.

C. Structural Equation Models

Structural equation models (SEMs) are another type of models (similar to PGMs) that is widely used to learn a directed acyclic graph (DAG) that encodes the conditional dependence of random variables [40]–[42]. Based on SEMs, the authors in [43] proposed a block coordinate descent algorithm to solve the joint optimisation problem of denoising the data and learning a directed graph. The joint learning framework is further extended to time series [44], where the structural vector autoregressive models (SVARMs) replace the linear SEMs to handle temporal dependency. The main difference from our work is that their denoising function is an identity mapping without side information as covariates. The work in [42] also considers the temporal dependency in learning a DAG with SVARMs, but does not consider the denoising scenario.

D. Graph Signal Processing

In the context of GSP, every observation on a collection of nodes is defined as a graph signal. GSP-based graph learning models have seen an increasing interest in the literature [10], [11] and further fall into two distinct branches. The first branch assumes graph signals are outcomes of diffusion processes on graphs and reconstructs a graph from signals according to the diffusion model [12]–[15]. The other branch constructs a graph by promoting the global smoothness of graph signals, which is defined by the Laplacian quadratic form [16], [17] or more generally via total variation [21]. Smoothness-based methods are related to GMRFs by recognising that the Laplacian quadratic form is closely related to the log-likelihood of the precision matrix defined as the graph Laplacian. Our work can be regarded as an extension to smoothness-based graph construction.

In the literature of smoothness-based GSP graph learning, the authors in [17], [21] adopt a two-step learning algorithm to learn an undirected graph while denoising graph signals. They simply assume an identity mapping between the actual graph signals and noisy observations, which is different from our work that considers side information. The most related work is proposed in [45], which uses kernel ridge regression with observation-side covariates to infer graph signals. However, their work mainly focuses on data prediction and graph learning is only a byproduct in their approach. In Section V-A, we will show, both theoretically and empirically, that their method uses a smoothness term that imprecisely incorporates the observation-side dependency in the learned graph structure, leading to an inferior performance in learning a graph.

In terms of the observation-side dependency, there exist some GSP graph learning models that consider temporal dependency in graph signals. A so-called spatiotemporal smoothness was proposed in [46], [47] to transform the graph signals using a temporally weighed difference operator. If every timestamp is equally important, the operator is equivalent to a prepossessing step to make the time series observed on each node stationary. It should be noted that there is another branch of research assuming that the temporal dependency in graph signals originates in the dynamic changes in the edges [48], [49], and therefore the problem is formulated as learning a dynamic series of graphs, which is different from the goal of our paper.

E. Graph Neural Networks

A new branch of graph learning models is developed from the perspective of GNNs. Essentially, GNNs discover the patterns in graph-structured data in a hierarchical manner [50]–[52]. The activations at intermediate layers, e.g. the l-th layer, can be interpreted as a new representation for the nodes in the embedding space that incorporates the information from a specifically defined neighbourhood of the nodes. The authors in [22], [53] thus defined the strength of connectivity between nodes i and j based on the pairwise similarity of their embeddings $h_i^{(l)}$ and $h_j^{(l)}$ at the l-th layer of the GNN architecture. The authors in [54] extended this method to construct a directed graph in the process of training a GNN that deals with time series data. The main goal of these methods is to improve the performance of node-related tasks (e.g. classification or prediction) and graph learning is only a byproduct, whose performance is often not guaranteed. The recent works in [23], [55]–[57] start to incorporate an additional loss for recovering graphs while training the GNNs. However, a significant limitation of most GNN-based methods is that they typically require a large volume of training data and the learned connectivity is often less explainable compared to PGM and GSP methods.

III. Preliminaries

A. Smoothness-Based Graph Learning in GSP

Observing a data matrix $\mathbf{Y} \in \mathbb{R}^{n \times m}$ whose ij-th entry y_{ij} corresponds to the observation on the j-th node in the i-th graph signal, we are interested in constructing an undirected and weighted graph $\mathcal{G} = \{\mathcal{V}, \mathcal{E}, \mathbf{W}\}$. The node set $\mathcal{V}$ represents a collection of variables, where $|\mathcal{V}| = m$. The edge set $\mathcal{E}$ represents the relational structure among them to be inferred. The structure is completely characterised by the weighted adjacency matrix $\mathbf{W}$ whose jj'-th entry is $w_{jj'}$. If two nodes j and j' are connected by an edge $e_{jj'} \in \mathcal{E}$ then $w_{jj'} > 0$, else if $e_{jj'} \notin \mathcal{E}$ then $w_{jj'} = 0$. The graph Laplacian matrix is defined as $\mathbf{L} = \text{diag}(\mathbf{W}\mathbf{1}) - \mathbf{W}$, where $\mathbf{1}$ denotes the all-one vector. $\mathbf{L}$ and $\mathbf{W}$ are equivalent and complete representations of the graph on a given set of nodes.

In the literature of GSP, one typical approach of constructing a graph from $\mathbf{Y}$ is formulated as minimising the variation of signals on graphs as measured by the Laplacian quadratic form[1] [7], [16], [17]:

$$\min_{\mathbf{L} \in \mathcal{L}} \text{Tr}(\mathbf{Y}\mathbf{L}\mathbf{Y}^\top) + \lambda \Omega(\mathbf{L}) \tag{1}$$

where $\mathcal{L} = \{\mathbf{L} | \mathbf{L}\mathbf{1} = 0, \mathbf{L}_{jj'} = \mathbf{L}_{j'j} \leq 0, \forall j \neq j'\}$ defines the space of valid Laplacian matrices, and $\Omega(\mathbf{L})$ is a regularisation term with a hyperparameter $\lambda > 0$. Equivalently, the problem can be formulated using the weighted adjacency matrix $\mathbf{W}$ such that

$$\min_{\mathbf{W} \in \mathcal{W}} \frac{1}{2} \sum_{i=1}^{n} \sum_{j,j'} w_{jj'} (y_{ij} - y_{ij'})^2 + \lambda \Omega(\mathbf{W}) \tag{2}$$

where $\mathcal{W} = \{\mathbf{W} | \text{diag}(\mathbf{W}) = \mathbf{1}, w_{jj'} = w_{jj'} \geq 0, \forall j \neq j'\}$ defines the space of valid weighted adjacency matrices. Popular choices of regularisation include the sum barrier $\Omega(\mathbf{L}) = ||\mathbf{L}||_F^2$ and $\Omega(\mathbf{L}) = |\text{Tr}(\mathbf{L}) - m|$ (often added as a constraint such that $\text{Tr}(\mathbf{L}) = m$) to prevent trivial solutions where all edge weights are zero and meanwhile controlling the variations of edge weights [17], or the log-barrier $\Omega(\mathbf{W}) = \mathbf{1}^\top \log(\mathbf{W}\mathbf{1})$ to prevent isolated nodes and promote connectivity [16].

With a fixed Frobenius norm for $\mathbf{Y}$, a small value of the objective in Eq. (1) implies that $\mathbf{Y}$ is smooth on $\mathcal{G}$ in the sense that neighbouring nodes have similar observations. The authors in [17] further propose a probabilistic generative model of the noise-free smooth observations $\mathbf{Y} = [\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_n]^\top$ such that

$$\mathbf{y}_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \mathbf{L}^\dagger), \quad i = 1, 2, \ldots, n \tag{3}$$

where $(\cdot)^\dagger$ denotes the Moore-Penrose pseudo-inverse of a matrix. This leads to a graph learning framework which solves an optimisation problem similar to Eq. (1).

B. Kronecker Product Kernel Regression

Taking a functional viewpoint on the generation of graph-structured data matrix, we can make use of the well-studied formalism of Kronecker product kernel ridge regression to infer the latent function [58]. Specifically, we consider $f : \mathcal{Z} \times \mathcal{X} \to \mathbb{R}$ to be an element of a reproducing kernel Hilbert space (RKHS) corresponding to the product kernel function $\kappa_\otimes = \kappa_\mathcal{Z} \otimes \kappa_\mathcal{X}$ on $\mathcal{Z} \times \mathcal{X}$.

A kernel function can be expressed as an inner product in a corresponding feature space, i.e. $\kappa_\mathcal{Z}(\mathbf{z}_i, \mathbf{z}_{i'}) = \langle \phi_\mathcal{Z}(\mathbf{z}_i), \phi_\mathcal{Z}(\mathbf{z}_{i'}) \rangle_{\mathcal{H}_\mathcal{Z}}$ where $\phi_\mathcal{Z} : \mathcal{Z} \to \mathcal{H}_\mathcal{Z}$ and $\kappa_\mathcal{X}(\mathbf{x}_i, \mathbf{x}_{i'}) = \langle \phi_\mathcal{X}(\mathbf{x}_i), \phi_\mathcal{X}(\mathbf{x}_{i'}) \rangle_{\mathcal{H}_\mathcal{X}}$, where $\phi_\mathcal{X} : \mathcal{X} \to \mathcal{H}_\mathcal{X}$. An explicit representation of feature maps $\phi_\mathcal{X}$ and $\phi_\mathcal{Z}$ is not necessary and the dimension of mapped feature vectors could be high and even infinite. By the representer theorem, the function f that fits the data $\mathbf{Y}$ takes the form

$$f(\mathbf{z}, \mathbf{x}) = \sum_{i=1}^{n} \sum_{j=1}^{m} a_{ij} \kappa_\mathcal{Z}(\mathbf{z}_i, \mathbf{z}) \kappa_\mathcal{X}(\mathbf{x}_j, \mathbf{x}) \tag{4}$$

where $a_{ij} \in \mathbb{R}$ are the coefficients to be learned, and the estimated value $\hat{y}_{ij} = f(\mathbf{z}_i, \mathbf{x}_j)$. Denoting the corresponding kernel matrices as $\mathbf{K}_z$ and $\mathbf{K}_x$, where $(\mathbf{K}_z)_{ii'} = \kappa_\mathcal{Z}(\mathbf{z}_i, \mathbf{z}_{i'})$ and $(\mathbf{K}_x)_{jj'} = \kappa_\mathcal{X}(\mathbf{x}_j, \mathbf{x}_{j'})$, we have the matrix form

$$\hat{\mathbf{Y}} = \mathbf{K}_z \mathbf{A} \mathbf{K}_x \tag{5}$$

where $\hat{\mathbf{Y}}$ is an approximation to $\mathbf{Y}$ and the coefficient matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ has the ij-th entry as a_{ij}. We assume the observation $\mathbf{Y}$ is a noisy version of $\hat{\mathbf{Y}}$, where the noise is *i.i.d.* normally distributed random variables such that $\epsilon_{ij} = \mathbf{Y}_{ij} - \hat{\mathbf{Y}}_{ij} \sim \mathcal{N}(0, \sigma_\epsilon^2)$, where σ_ϵ^2 measures the noise level. This leads to a natural choice of the sum of squared errors as the loss function.

A standard Tikhonov regulariser is often added to the regression model to reduce overfitting and penalise complex functions, which is defined in our case as

$$\begin{aligned} ||f||_{\mathcal{H}_\otimes}^2 &= \text{vec}(\mathbf{A})^\top (\mathbf{K}_x \otimes \mathbf{K}_z) \text{vec}(\mathbf{A}) \\ &= \text{Tr}(\mathbf{K}_z \mathbf{A} \mathbf{K}_x \mathbf{A}^\top) \end{aligned} \tag{6}$$

where $\text{vec}(\cdot)$ is the vectorisation operator for a matrix. We now arrive at the following optimisation problem to infer the function $f(\mathbf{z}, \mathbf{x})$ that approximates the observation matrix $\mathbf{Y}$ such that

$$\min_{\mathbf{A}} ||\mathbf{Y} - \mathbf{K}_z \mathbf{A} \mathbf{K}_x||_F^2 + \lambda \text{Tr}(\mathbf{K}_z \mathbf{A} \mathbf{K}_x \mathbf{A}^\top) \tag{7}$$

where the hyperparameter $\lambda > 0$ controls the penalisation of the complexity of the function to be learned.

To have a better understanding of how this model is expressive for the two-side dependency, we show that the objective in Eq. (7) can be derived from a Bayesian viewpoint. In the vector form, i.e. $\mathbf{a} = \text{vec}(\mathbf{A})$ and $\mathbf{y} = \text{vec}(\mathbf{Y})$, we assume that both the data likelihood $p(\mathbf{y}|\mathbf{a})$ and the prior $p(\mathbf{a})$ follow a Gaussian

[1] We acknowledge that the conventional form of the Laplacian quadratic in GSP literature is $\text{Tr}(\mathbf{Y}^\top \mathbf{L}\mathbf{Y})$, where each column of $\mathbf{Y}$ corresponds to a graph signal. In our case, $\mathbf{Y}$ has two-side dependency such that either a column or a row may be regarded as a graph signal. The term $\text{Tr}(\mathbf{Y}\mathbf{L}\mathbf{Y}^\top)$ measures the smoothness of row vectors over a column graph. This formulation is however consistent with the statistical modelling convention where each column in $\mathbf{Y}$ is often regarded as a random variable and the graph of main interest is the column graph.

distribution:

$$\mathbf{y}|\mathbf{a} \sim \mathcal{N}((\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a}, \sigma_\epsilon^2 \mathbf{I}_{nm}), \tag{8a}$$

$$\mathbf{a} \sim \mathcal{N}(\mathbf{0}_{nm}, \mathbf{K}_x^\dagger \otimes \mathbf{K}_z^\dagger), \tag{8b}$$

where $\mathbf{0}_{mn}$ is a zero-vector of length mn. Notice that $\mathbf{K}_x$ and $\mathbf{K}_z$ (and their Kronecker product) can be either singular or non-singular matrices, depending on the kernel choice. For the sake of simplicity, we use the pseudo-inverse notation throughout the paper. Now, the marginal likelihood of $\mathbf{y}$ is

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}_{mn}, \mathbf{K}_x \otimes \mathbf{K}_z + \sigma_\epsilon^2 \mathbf{I}_{nm}) \tag{9}$$

from which we can see that the covariance structure of $\mathbf{y}$ can be understood as a combination of $\mathbf{K}_x$ and $\mathbf{K}_z$. Specifically, in the noise-free scenario where $\sigma_\epsilon^2 = 0$, the covariance matrix over the rows of $\mathbf{Y}$ is

$$\text{Cov}_r[\mathbf{Y}] = E[\mathbf{Y}^\top \mathbf{Y}] = \text{Tr}(\mathbf{K}_z)\mathbf{K}_x \tag{10}$$

Similarly, the covariance matrix over the columns of $\mathbf{Y}$ is

$$\text{Cov}_c[\mathbf{Y}] = E[\mathbf{Y}\mathbf{Y}^\top] = \text{Tr}(\mathbf{K}_x)\mathbf{K}_z \tag{11}$$

The proof for Eq. (10) and Eq. (11) can be found in [59]. In Appendix A, we show that the maximisation of the log-posterior of the coefficient vector $\mathbf{a}$ leads to the objective in Eq. (7).

IV. Kernel Graph Learning

A. A Smoothness Measure for Dependent Graph Signals

Given n dependent graph signals observed on m nodes, we want to infer a graph Laplacian matrix $\mathbf{L} \in \mathbb{R}^{m \times m}$ that captures the invariant relationship between node pairs. The invariance here is understood in the sense that $\mathbf{L}$ does not change due to the observation-side dependency. This is motivated by the fact that the classical Laplacian quadratic in Eq. (1) as a smoothness measure is no longer applied for non-$i.i.d.$ signals. By recognising a Kronecker product covariance structure in a noise-free version of $\mathbf{y}$ from Eq. (9), we define a novel measure of smoothness for the graph signals with observation-side dependency over an invariant graph (with graph Laplacian $\mathbf{L}$) as

$$\mathbf{y}^\top (\mathbf{L} \otimes \mathbf{K}_z^\dagger)\mathbf{y} \tag{12}$$

where $\mathbf{y}$ is modelled by the Kronecker product kernel regression in Section III-B and thus the measure can be approximated, based on Eq. (5), by

$$\begin{aligned} \hat{\mathbf{y}}^\top (\mathbf{L} \otimes \mathbf{K}_z^\dagger)\hat{\mathbf{y}} &= \text{vec}(\hat{\mathbf{Y}})^\top (\mathbf{L} \otimes \mathbf{K}_z^\dagger)\text{vec}(\hat{\mathbf{Y}}) \\ &= \text{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z) \end{aligned} \tag{13}$$

One interpretation of the above smoothness term is associated with a Kronecker product graph, if we further assume the observation-side dependency corresponds to a graph such that $\mathbf{K}_z^\dagger = \mathbf{L}_z$. Such an assumption is typically used in defining kernel functions on graphs [60]. Now, Eq. (12) turns into a standard Laplacian quadratic form such that

$$\mathbf{y}^\top (\mathbf{L} \otimes \mathbf{K}_z^\dagger)\mathbf{y} = \mathbf{y}^\top \mathbf{L}_\otimes \mathbf{y} \tag{14}$$

where $\mathbf{L}_\otimes = \mathbf{L} \otimes \mathbf{L}_z$. We further show in Appendix B that $\mathbf{L}_\otimes$ is a Laplacian-like operator on which a notion of frequency of $\mathbf{y}$ in the context of graph Fourier transform can be defined.

Furthermore, $\mathbf{L}_\otimes$ brings another interpretation for the smoothness term. It can be viewed as a Laplacian-based regulariser which can be added to the problem of inferring a function f that fits the graph signals in Eq. (7):

$$\begin{aligned} ||f||^2_{\mathcal{H}_\mathcal{M}} &= \langle f, \mathbf{L}_\otimes f \rangle_{\mathcal{H}_\mathcal{M}} \\ &= \langle f, (\mathbf{L} \otimes \mathbf{K}_z^\dagger) f \rangle_{\mathcal{H}_\mathcal{M}} \\ &= \text{vec}(\hat{\mathbf{Y}})^\top (\mathbf{L} \otimes \mathbf{K}_z^\dagger)\text{vec}(\hat{\mathbf{Y}}) \\ &= \text{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z) \end{aligned} \tag{15}$$

where $\mathcal{M}$ denotes a compact manifold.[2]

The difference between $\mathbf{K}_x$ and $\mathbf{L}$ is worth clarifying, as they both represent the relationship between nodes in the graph. $\mathbf{K}_x$ is obtained from node-side covariates $\mathbf{x} \in \mathcal{X}$ and can be viewed as an initial estimate of the dependency between nodes obtained from prior information, e.g. node feature in addition to graph signals. By contrast, $\mathbf{L}$ can be interpreted as the refined estimate, i.e. the graph of interest, obtained by learning from the observed graph signals.

B. Learning Framework

We propose a joint learning framework for inferring the function f that fits the graph signals as in Eq. (7) as well as the underlying graph $\mathbf{L}$ to capture the relationship between the nodes as in Eq. (15). This relationship is disentangled from the observation-side dependency of non-$i.i.d.$ graph signals with the notion of smoothness introduced in Section IV-A. We name this framework Kernel Graph Learning (**KGL**) which aims at solving the following problem:

$$\begin{aligned} \min_{\mathbf{L} \in \mathcal{L}, \mathbf{A}} \; J(\mathbf{L}, \mathbf{A}) &= ||\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x||_F^2 + \lambda \text{Tr}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x\mathbf{A}^\top) \\ &\quad + \rho \text{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z) + \psi ||\mathbf{L}||_F^2 \\ \text{s.t.} \quad \text{Tr}(\mathbf{L}) &= m \end{aligned} \tag{16}$$

where $\mathcal{L} = \{\mathbf{L} | \mathbf{L}\mathbf{1} = 0, \mathbf{L}_{jj'} = \mathbf{L}_{j'j} \leq 0, \forall j \neq j'\}$, and $||\cdot||_F$ denotes the Frobenius norm. The first two terms correspond to the functional learning part where the hyperparameter $\lambda > 0$ controls the complexity of the function f for fitting $\mathbf{Y}$. The last two terms and the constraints can be viewed as a graph learning model in Eq. (1) with the fitted values of $\mathbf{Y}$ as input graph signals and the sum barrier as the graph regulariser. The hyperparameter $\rho > 0$ controls the relative importance between fitting the function and learning the graph, and $\psi > 0$ controls the distribution of edge weights. The trace constraint acts as a normalisation term such that the sum of learned edge weights equals the number of nodes. The model is compatible with constraints that enforce other properties on the learned graph, e.g. the log barrier introduced in Section III-A. This paper is

[2] We refer the interested reader to [61] for the theorem of manifold regularisation with the Laplace-Beltrami operator.

mainly based on one of the choices for the constraints in order to maintain focus on the general framework.

C. Optimisation: Alternating Minimisation

We first recognise that Eq. (16) is a biconvex optimisation problem, i.e. convex w.r.t $\mathbf{A}$ while $\mathbf{L}$ is fixed and vice versa. This motivates an iterative block-coordinate descent algorithm that alternates between minimisation in $\mathbf{A}$ and $\mathbf{L}$ [62], [63]. In this section, we derive the update steps of $\mathbf{A}$ and $\mathbf{L}$ separately, propose the main algorithm in Algorithm 1, and prove its convergence.

1) Update of $\mathbf{A}$*:* The update of coefficients $\mathbf{A}$ can be regarded as solving a Laplacian-regularised kernel regression [61]. Given $\mathbf{L}$, the optimisation problem of Eq. (16) becomes

$$\begin{aligned}\min_{\mathbf{A}} \ & ||\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x||_F^2 + \lambda \mathrm{Tr}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x\mathbf{A}^\top) \\ & + \rho \mathrm{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z)\end{aligned} \tag{17}$$

and, after dropping constant terms,

$$\begin{aligned}\min_{\mathbf{A}} \ & J_{\mathbf{L}}(\mathbf{A}) = \mathrm{Tr}(\mathbf{A}^\top\mathbf{K}_z^2\mathbf{A}\mathbf{K}_x^2) - 2\mathrm{Tr}(\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z\mathbf{Y}) \\ & + \lambda\mathrm{Tr}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x\mathbf{A}^\top) + \rho\mathrm{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z).\end{aligned} \tag{18}$$

Denote $\mathbf{a} = \mathrm{vec}(\mathbf{A})$ and $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$, we obtain a dual-form for $J_{\mathbf{L}}(\mathbf{A})$ such that

$$\begin{aligned}J_{\mathbf{L}}(\mathbf{a}) &= \mathbf{a}^\top(\mathbf{K}_x^2 \otimes \mathbf{K}_z^2)\mathbf{a} - 2\mathbf{a}^\top(\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{y} \\ &\quad + \lambda\mathbf{a}^\top(\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a} + \rho\mathbf{a}^\top\left((\mathbf{K}_x\mathbf{L}\mathbf{K}_x) \otimes \mathbf{K}_z\right)\mathbf{a} \\ &= \mathbf{a}^\top\left(\mathbf{K}^2 + \lambda\mathbf{K} + \rho\mathbf{S} \otimes \mathbf{K}_z\right)\mathbf{a} - 2\mathbf{a}^\top\mathbf{K}\mathbf{y}\end{aligned} \tag{19}$$

where $\mathbf{K} = \mathbf{K}_x \otimes \mathbf{K}_z$, and $\mathbf{S} = \mathbf{K}_x\mathbf{L}\mathbf{K}_x$ for simplicity. We prove in Appendix C that $\mathbf{K}^2 + \lambda\mathbf{K} + \rho\mathbf{S} \otimes \mathbf{K}_z$ is positive semi-definite thus Eq. (19) is an unconstrained quadratic programme. The gradient of $J_{\mathbf{L}}(\mathbf{a})$ w.r.t. $\mathbf{a}$ is

$$\nabla J_{\mathbf{L}}(\mathbf{a}) = \left(\mathbf{K}^2 + \lambda\mathbf{K} + \rho\mathbf{S} \otimes \mathbf{K}_z\right)\mathbf{a} - \mathbf{K}\mathbf{y}. \tag{20}$$

Strictly speaking, the matrix $\mathbf{K}^2 + \lambda\mathbf{K} + \rho\mathbf{S} \otimes \mathbf{K}_z$ may contain zero eigenvalues which makes it not invertible. However, a majority of popular kernel functions for $\mathbf{K}_x$ and $\mathbf{K}_z$ are positive-definite, e.g. the RBF kernel. Since Kronecker product preserves positive definiteness, $\mathbf{K}$ and hence the whole matrix is positive-definite and invertible. Setting $\nabla J_{\mathbf{L}}(\mathbf{a}) = \mathbf{0}$ and cancelling out $\mathbf{K}$, we have:

$$\begin{aligned}&\left(\mathbf{K}^2 + \lambda\mathbf{K} + \rho\mathbf{S} \otimes \mathbf{K}_z\right)\mathbf{a} - \mathbf{K}\mathbf{y} = \mathbf{0} \\ \Rightarrow\ &\left(\mathbf{K}^2 + \lambda\mathbf{K} + \rho\mathbf{K}(\mathbf{L}\mathbf{K}_x \otimes \mathbf{I}_n)\right)\mathbf{a} - \mathbf{K}\mathbf{y} = \mathbf{0} \\ \Rightarrow\ &\left(\mathbf{K} + \lambda\mathbf{I}_{mn} + \rho\mathbf{L}\mathbf{K}_x \otimes \mathbf{I}_n\right)\mathbf{a} = \mathbf{y}\end{aligned} \tag{21}$$

Denote $\mathbf{H} = \mathbf{K} + \lambda\mathbf{I}_{mn} + \rho\mathbf{L}\mathbf{K}_x \otimes \mathbf{I}_n$, where $\mathbf{H}$ has a dimension of $nm \times nm$. We can therefore obtain a closed-form solution such that

$$\mathbf{a} = \mathbf{H}^{-1}\mathbf{y} \tag{22}$$

where the inverse of $\mathbf{H}$ requires $\mathcal{O}(n^3\ m^3)$.

To further reduce the complexity, we make use of the Kronecker structure and matrix tricks. We first recognise $\mathbf{H}$ as

$$\mathbf{H} = \left((\rho\mathbf{L} + \lambda\mathbf{K}_x^{-1}) \oplus \mathbf{K}_z\right)(\mathbf{K}_x \otimes \mathbf{I}_n)$$

where $\oplus$ is the Kronecker sum. With the eigendecomposition $\mathbf{K}_x = \mathbf{Q}_x\mathbf{\Lambda}_x\mathbf{Q}_x^\top$, $\mathbf{K}_z = \mathbf{Q}_z\mathbf{\Lambda}_z\mathbf{Q}_z^\top$ and $\rho\mathbf{L} + \lambda\mathbf{K}_x^{-1} = \mathbf{U}_x\mathbf{D}_x\mathbf{U}_x^\top$, we have

$$\mathbf{H} = (\mathbf{U}_x \otimes \mathbf{Q}_z)(\mathbf{D}_x \oplus \mathbf{\Lambda}_z)\left(\mathbf{U}_x^\top \otimes \mathbf{Q}_z^\top\right)\left(\mathbf{Q}_x\mathbf{\Lambda}_x\mathbf{Q}_x^\top \otimes \mathbf{I}_n\right). \tag{23}$$

Here, $\mathbf{D}_x \oplus \mathbf{\Lambda}_z$ is an $mn \times mn$ diagonal matrix with entries being all the pairwise sums of eigenvalues in $\mathbf{D}_x$ and $\mathbf{\Lambda}_z$. Inversion of that matrix is thus $O(mn)$. We can now obtain cheap inversion with

$$\begin{aligned}\mathbf{H}^{-1}\mathbf{y} = &\left(\mathbf{Q}_x\mathbf{\Lambda}_x^{-1}\mathbf{Q}_x^\top \otimes \mathbf{I}_n\right)(\mathbf{U}_x \otimes \mathbf{Q}_z)(\mathbf{D}_x \oplus \mathbf{\Lambda}_z)^{-1} \\ &\cdot \mathrm{vec}\left(\mathbf{Q}_z^\top\mathbf{Y}\mathbf{U}_x\right).\end{aligned} \tag{24}$$

The operation $(\mathbf{D}_x \oplus \mathbf{\Lambda}_z)^{-1}\mathrm{vec}(\mathbf{Q}_z^\top\mathbf{Y}\mathbf{U}_x)$ is simply rescaling each term in the mn-vector with the corresponding diagonal entry of $(\mathbf{D}_x \oplus \mathbf{\Lambda}_z)^{-1}$. If we denote $\mathbf{d}_x$ and $\mathbf{d}_z$ column vectors containing the diagonal entries this can be expressed as $\mathrm{vec}(\mathbf{B})$ with

$$\mathbf{B} = \left(\mathbf{1}_n\mathbf{d}_x^\top + \mathbf{d}_z\mathbf{1}_m^\top\right)^{\circ-1} \circ \left(\mathbf{Q}_z^\top\mathbf{Y}\mathbf{U}_x\right),$$

where $\circ$ denotes the Hadamard product and $(\cdot)^{\circ-1}$ denotes entrywise inversion. Remaining operations are now direct and give the closed-form solution as

$$\begin{aligned}\mathbf{A} &= \mathbf{Q}_z\mathbf{B}\mathbf{U}_x^\top\mathbf{Q}_x\mathbf{\Lambda}_x^{-1}\mathbf{Q}_x^\top \\ &= \mathbf{Q}_z\mathbf{B}\mathbf{U}_x^\top\mathbf{K}_x^{-1} \\ &= \mathbf{Q}_z\left[\left(\mathbf{1}_n\mathbf{d}_x^\top + \mathbf{d}_z\mathbf{1}_m^\top\right)^{\circ-1} \circ \left(\mathbf{Q}_z^\top\mathbf{Y}\mathbf{U}_x\right)\right]\mathbf{U}_x^\top\mathbf{K}_x^{-1}.\end{aligned} \tag{25}$$

Notice that this solution requires only matrix multiplications and inversions and eigendecompositions on $m \times m$ or $n \times n$ matrices, giving an overall computational cost of $\mathcal{O}(n^3 + m^3 + nm^2 + n^2m)$.

When $\mathbf{K}_x$ and $\mathbf{K}_z$ are not invertible, we suggest using the gradient descent to avoid the inverse of a large matrix of dimension $nm \times nm$. The update step using Eq. (20) is:

$$\mathbf{a}^{(\tau+1)} = \mathbf{a}^{(\tau)} - \gamma\nabla J_{\mathbf{L}}(\mathbf{a}^{(\tau)}) \tag{26}$$

where $\gamma > 0$ is the step size.

2) Update of $\mathbf{L}$*:* Given $\mathbf{A}$, the optimisation problem of Eq. (16) becomes

$$\begin{aligned}\min_{\mathbf{L}\in\mathcal{L}} \ & J_{\mathbf{A}}(\mathbf{L}) = \rho\mathrm{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z) + \psi||\mathbf{L}||_F^2 \\ \text{s.t} \ & \mathrm{Tr}(\mathbf{L}) = m\end{aligned} \tag{27}$$

which is a constrained quadratic programme w.r.t. $\mathbf{L}$. By taking $\mathbf{P} = \mathbf{K}_z^{1/2}\mathbf{A}\mathbf{K}_x$, the problem fits in the learning framework in Eq. (1). We use the package CVXPY [64] to solve this problem.

The overall KGL framework is presented in Algorithm 1. The convergence for each update step of $\mathbf{A}^{(t)}$ and $\mathbf{L}^{(t)}$ is guaranteed in solving the respective convex optimisation of Eq. (18) and Eq. (27). It should be noted that the step size γ in Eq. (26) needs to be set appropriately for the gradient descent to converge.

Algorithm 1: Kernel Graph Learning (**KGL**).

```
Input: Observation $\mathbf{Y}$, node-side kernel matrix $\mathbf{K}_x$,
       observation-side kernel matrix $\mathbf{K}_z$,
       hyper-parameters $\rho$, $\lambda$ and $\psi$, tolerance level $\epsilon$.
Initialisation: $t = 0$, $\mathbf{A} = \mathbf{0} \in \mathbb{R}^{n \times m}$
while $|\mathbf{L}^{(t)} - \mathbf{L}^{(t-1)}| > \epsilon$ and $|\mathbf{A}^{(t)} - \mathbf{A}^{(t-1)}| > \epsilon$ do
    Update $\mathbf{L}^{(t)} = \arg\min J_{\mathbf{A}^{(t-1)}}(\mathbf{L})$ by solving Eq. (27)
    Update $\mathbf{A}^{(t)} = \arg\min J_{\mathbf{L}^{(t)}}(\mathbf{A})$ by
      (i) using the closed-form solution in Eq. (25), if $\mathbf{K}_x$
      and $\mathbf{K}_z$ are invertible; or
      (ii) updating vec($\mathbf{A}^{(t)}$) with gradient descent in
      Eq. (26), otherwise
    $t = t + 1$
end while
return $\mathbf{L}^{(t)}$, $\mathbf{A}^{(t)}$
```

We suggest a $\gamma \leq 10^{-4}$ from empirical results. The choices of hyperparameters λ, ρ and ψ are discussed in Section V-F. We now provide the following statement on convergence of Algorithm 1.

Lemma IV.1: The sequence $\{J(\mathbf{L}^{(t)}, \mathbf{A}^{(t)})\}$ generated by Algorithm 1 converges monotonically and the solution obtained by Algorithm 1 is a stationary point of Eq. (16).

Proof: We follow the convergence results of the alternate convex search in [63] and that of a more general cyclic block-coordinate descent algorithm in [62]. By recognising that Eq. (18) and Eq. (27) are quadratic programmes (with Lemma C.1 in Appendix C), the problem of Eq. (16) is a bi-convex problem with all the terms differentiable and the function $J(\mathbf{L}, \mathbf{A})$ continuous and bounded from below. Theorem 4.5 in [63] states that the sequence $\{J(\mathbf{L}^{(t)}, \mathbf{A}^{(t)})\}$ generated by Algorithm 1 converges monotonically. Theorem 4.1 in [62] states that the sequence $\{\mathbf{L}^{(t)}\}$ and $\{\mathbf{A}^{(t)}\}$ generated by Algorithm 1 are defined and bounded. Furthermore, according to Theorem 5.1 in [62], every cluster point $\{\mathbf{L}^{(t)}, \mathbf{A}^{(t)}\}$ is a coordinatewise minimum point of J hence the solution is a stationary point of Eq. (16). ■

Our empirical results suggest that after only 10 iterations or less, the sequence $\{\mathbf{L}^{(t)}, \mathbf{A}^{(t)}\}$ does not change more than the tolerance level. The computational complexity of **KGL** in Algorithm 1 is dominated by the step of updating $\mathbf{A}$. It requires $\mathcal{O}(n^3 + m^3 + nm^2 + n^2\ m)$ to compute the closed-form solution of $\mathbf{A}$ or $\mathcal{O}(n^3\ m^3)$ to compute the gradient in Eq. (20) if the closed-form solution is not applied when $\mathbf{K}_z$ and $\mathbf{K}_x$ are not invertible. Updating $\mathbf{L}$ requires $\mathcal{O}(m^2)$. Overall, for T iterations that guarantee the convergence of Algorithm 1, it requires either $\mathcal{O}(T(n^3 + m^3 + nm^2 + n^2\ m))$ or $\mathcal{O}(T(n^3\ m^3))$ operations, depending on whether $\mathbf{K}_z$ and $\mathbf{K}_x$ are chosen to be invertible. We note that one could readily appeal to large-scale kernel approximation methods for further reduction of computational and storage complexity of the **KGL** framework, and hence broaden its applicability to larger datasets. There are two main approaches to large-scale kernel approximations and both can be applied to **KGL**. The former focuses on kernel matrix approximations using methods such as Nyström sampling [65], while the latter deals with the approximation of the kernel function itself, using methods such as Random Fourier Features [66]. Nonetheless, this paper focuses on the modelling perspective, and we will leave the algorithmic improvement as a future direction.

D. Special Cases of Kernel Graph Learning

Independent observations. It is often assumed that graph signals are *i.i.d.* hence there exists no dependency along the observation side. This is equivalent to setting $\mathbf{K}_z = \mathbf{I}_n$ in our framework. We refer to this special case of **KGL** as Node-side Kernel Graph Learning (**KGL-N**):

$$\begin{aligned} \min_{\mathbf{L}\in\mathcal{L},\mathbf{A}} \ & ||\mathbf{Y} - \mathbf{A}\mathbf{K}_x||_F^2 + \lambda \text{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{A}^\top) \\ & + \rho\text{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top) + \psi||\mathbf{L}||_F^2 \\ \text{s.t} \ & \text{Tr}(\mathbf{L}) = m \end{aligned} \tag{28}$$

No node-side information. It also may be the case that no node-side information is available for the problem at hand. In this case, we can simply set $\mathbf{K}_x = \mathbf{I}_m$ in **KGL**, leading to to Observation-side Kernel Graph Learning (**KGL-O**):

$$\begin{aligned} \min_{\mathbf{L}\in\mathcal{L},\mathbf{A}} \ & ||\mathbf{Y} - \mathbf{K}_z\mathbf{A}||_F^2 + \lambda \text{Tr}(\mathbf{A}^\top\mathbf{K}_z\mathbf{A}) \\ & + \rho\text{Tr}(\mathbf{A}\mathbf{L}\mathbf{A}^\top\mathbf{K}_z) + \psi||\mathbf{L}||_F^2 \\ \text{s.t} \ & \text{Tr}(\mathbf{L}) = m \end{aligned} \tag{29}$$

In the cases of one-side **KGL**, the optimisation again follows the alternating minimisation, i.e. to solve for **KGL-N** or **KGL-O**), one can simply set $\mathbf{K}_z = \mathbf{I}_n$ or $\mathbf{K}_x = \mathbf{I}_m$ in Algorithm 1. It should be noted, however, that the update step of $\mathbf{A}$ requires less computational cost when either $\mathbf{K}_z = \mathbf{I}_n$ or $\mathbf{K}_x = \mathbf{I}_m$. Indeed, the objective function of $\mathbf{KGL} - \mathbf{N}$ can be decomposed into the sum according to n functions such that

$$J_{\mathbf{L}}(\{\mathbf{a}_i\}_{i=1}^n) = \sum_{i=1}^{n} \left(||\mathbf{y}_i - \mathbf{K}_x\mathbf{a}_i||_2^2 + \mathbf{a}_i^\top(\lambda\mathbf{K}_x + \rho\mathbf{S})\mathbf{a}_i \right) \tag{30}$$

where $\mathbf{A} = [\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n]^\top$ and $\mathbf{S} = \mathbf{K}_x\mathbf{L}\mathbf{K}_x$. Consequently, the update step can be parallelised.

E. Learning With Missing Observations

By modifying the least-squares loss in **KGL**, we propose an extension to jointly learn the underlying graph and function from graph-structured data with missing values. We encode the positions of missing values with a mask matrix $\mathbf{M} \in \mathbb{R}^{n \times m}$ such that $\mathbf{M}_{ij} = 0$ if $\mathbf{Y}_{ij}$ is missing, and $\mathbf{M}_{ij} = 1$ otherwise. Now, we only need to minimise the least-squares loss over observed $\mathbf{Y}_{ij}$ in the functional learning part, which leads to the formulation:

$$\begin{aligned} \min_{\mathbf{L}\in\mathcal{L},\mathbf{A}} \ & ||\mathbf{M} \circ (\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x)||_F^2 + \lambda \text{Tr}(\mathbf{K}_{\mathcal{Z}}\mathbf{A}\mathbf{K}_x\mathbf{A}^\top) \\ & + \rho\text{Tr}(\mathbf{A}\mathbf{K}_x\mathbf{L}\mathbf{K}_x\mathbf{A}^\top\mathbf{K}_z) + \psi||\mathbf{L}||_F^2 \\ \text{s.t} \ & \text{Tr}(\mathbf{L}) = m. \end{aligned} \tag{31}$$

This formulation also applies to one-side kernel graph learning, i.e. **KGL-N** or **KGL-O**, with $\mathbf{K}_z = \mathbf{I}_n$ or $\mathbf{K}_x = \mathbf{I}_m$.

The optimisation problem in Eq. (31) is a bi-convex problem and alternating minimisation can still be applied. The update step of $\mathbf{L}$ remains the same as in Eq. (27), but the gradient in the update step of $\mathbf{a} = \text{vec}(\mathbf{A})$ (Step 4. in Algorithm 1) becomes

$$\begin{aligned}\nabla J_{\mathbf{L}}(\mathbf{a}) = \; & (\mathbf{K}\text{diag}(\mathbf{m})\mathbf{K} + \lambda \mathbf{K} \\ & + \rho \mathbf{S} \otimes \mathbf{K}_z)\, \mathbf{a} - \mathbf{K}\text{vec}(\mathbf{M} \circ \mathbf{Y}) \qquad (32)\end{aligned}$$

where $\mathbf{m} = \text{vec}(\mathbf{M})$. The detailed derivation of the gradient is provided in Appendix D. We further assume $\mathbf{K}$ is invertible, which is a mild assumption as we have many choices of kernel functions for $\mathbf{K}_x$ and $\mathbf{K}_z$ to be invertible. Also noting $\mathbf{S} \otimes \mathbf{K}_z = \mathbf{K}(\mathbf{L}\mathbf{K}_x \otimes \mathbf{I}_n)$, the gradient becomes

$$\begin{aligned}\nabla J_{\mathbf{L}}(\mathbf{a}) = \; & (\text{diag}(\mathbf{m})\mathbf{K} + \lambda \mathbf{I}_{nm} + \rho(\mathbf{L}\mathbf{K}_x \otimes \mathbf{I}_n))\, \mathbf{a} \\ & - \text{vec}(\mathbf{M} \circ \mathbf{Y}). \qquad (33)\end{aligned}$$

One can either derive a close-form solution or use gradient descent based on Eq. (33).

V. Synthetic Experiments

A. General Settings

Groundtruth Graphs. Random graphs of m nodes are drawn from the Erdös-Rényi (ER), Barabási-Albert (BA) and stochastic block model (SBM) as groundtruth, which are denoted as $\mathcal{G}_{\text{ER}}$, $\mathcal{G}_{\text{BA}}$ and $\mathcal{G}_{\text{SBM}}$, respectively. The parameters of each network model are chosen to yield an edge density of 0.3. The edge weights are randomly drawn from a uniform distribution $\mathbf{W}_{ij} \sim \mathcal{U}(0,1)$. The weighted adjacency matrix is set as $\mathbf{W} = (\mathbf{W} + \mathbf{W}^\top)/2$ for symmetry and normalised such that the sum of edge weights is equal to m for ease in comparison. The graph Laplacian $\mathbf{L}$ is calculated from $\mathbf{L} = \text{diag}(\mathbf{W}\mathbf{1}) - \mathbf{W}$.

Groundtruth Data. We generate mild noisy data $\mathbf{Y} \in \mathbb{R}^{n \times m}$ from $\mathbf{Y} = \mathbf{K}_z \mathbf{A} \mathbf{K}_x + \mathbf{E}$, where $\mathbf{a} = \text{vec}(\mathbf{A})$ is drawn from $\mathbf{a} \sim \mathcal{N}(\mathbf{0}_{mn}, \mathbf{K}_x^\dagger \otimes \mathbf{K}_z^\dagger)$ according to Eq. (8b). Every entry of the noise matrix $\mathbf{E}$ is an $i.i.d.$ sample from $\mathbf{E}_{ij} \sim \mathcal{N}(0, \sigma_\epsilon^2)$. To test the proposed model against different levels of noises, we vary the value of σ_ϵ^2 in Section V-B. For all other synthetic experiments, we add a mild-level noise with $\sigma_\epsilon^2 = 0.01$. We choose $\mathbf{K}_x = (\mathbf{I} + \lambda \mathbf{L})^{-1}$, as it is a popular method to generate smooth signals in related work [16], [45]. We consider both dependence and independence along the observation side:

- **Independent Data**: $\mathbf{K}_z = \mathbf{I}_n$;
- **Dependent Data**: $\mathbf{K}_z$ is obtained from an RBF kernel evaluated on synthetic observation-side information $\mathbf{z} = [0, 1, 2, \ldots, n-1]^\top$, which can be interpreted as the time-stamps of a discrete-time Markov chain. The bandwidth parameter is chosen according to the median heuristic [67].

Model Candidates. To have a fair comparison, the models are divided into two groups. The first group contains the baseline models that cannot deal with observation-side dependence:

- **GL** (Eq. (14) in [16]): the GSP graph learning model in Eq. (1) with $\Omega(\mathbf{L}) = ||\mathbf{L}||_F^2$.
- **GL-2step** (Eq. (16) in [17]): a two-step GSP graph learning framework with an identity mapping as denoising function. From a modelling perspective, Eq. (13) in [21] proposed a similar model with more constraints on edge weights and a different optimisation algorithm. We treat them as the same kind of techniques.
- **KGL-N** (proposed model in Eq. (28)): $\mathbf{K}_z = \mathbf{I}_n$ in **KGL**.

The second group is examined with observation-side dependent data:

- **KGL-Agnostic** (Eq. (18) in [45]): As discussed in Section II, the joint learning model in [45] considered the observation-side kernel, but did not use the observation-side dependence in learning the graph. We denote their model as **KGL-Agnostic** with our notations:

$$\begin{aligned}\min_{\mathbf{L} \in \mathcal{L}, \mathbf{A}} \; & ||\mathbf{Y} - \mathbf{K}_z \mathbf{A}||_F^2 + \lambda \text{Tr}(\mathbf{A}^\top \mathbf{K}_z \mathbf{A}) \\ & + \rho \text{Tr}(\mathbf{K}_z \mathbf{A} \mathbf{L} \mathbf{A}^\top \mathbf{K}_z) + \psi ||\mathbf{L}||_F^2 \qquad (34)\end{aligned}$$

- **KGL** (proposed model in Eq. (16)): the main learning framework.
- **KGL-O** (proposed model in Eq. (29)): To have a fair comparison to **KGL-Agnostic**, we also assume the graph is agnostic to the model (i.e. $\mathbf{K}_x = \mathbf{I}_m$ as model input).

For each method, we determine the hyperparameters via a grid search, and report the highest performance achieved by the best set of hyperparameters.

Evaluation Metrics. Average precision score (APS) and normalised sum of squared errors ($\text{SSE}_{\mathcal{G}}$) are used to evaluate the graph estimates, and out-of-sample mean squared error (MSE_y) is used to evaluate the estimated entries of graph-structured data matrix that were not observed (or missing). The APS is defined in a binary classification scenario for graph structure recovery, which automatically varies the threshold of weights above which the edges are declared as learned edges. An APS score of 1 indicates that the algorithm can precisely detect the ground-truth edges and non-edges. The $\text{SSE}_{\mathcal{G}}$ is defined over learned adjacency matrix $\hat{\mathbf{W}}$ and the groundtruth adjacency matrix $\mathbf{W}_0$ as

$$\text{SSE}_{\mathcal{G}} = \frac{||\hat{\mathbf{W}} - \mathbf{W}_0||_F^2}{||\mathbf{W}_0||_F^2}.$$

The out-of-sample MSE_y of data matrix is defined with a mask matrix $\mathbf{M}$ (same as in Eq. (31)), where $\mathbf{M}_{ij} = 0$ indicates $\mathbf{Y}_{ij}$ is a missing entry:

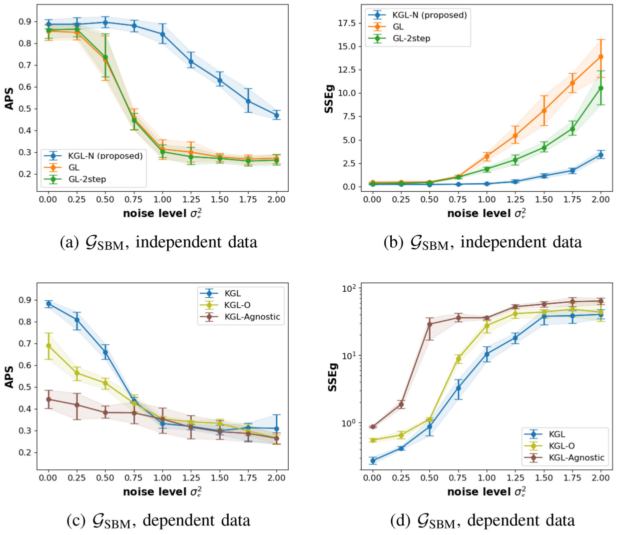
$$\text{Out-of-sample MSE}_y = \frac{||(\mathbf{1}\mathbf{1}^\top - \mathbf{M}) \circ (\hat{\mathbf{Y}} - \mathbf{Y})||_F^2}{||\mathbf{1}\mathbf{1}^\top - \mathbf{M}||_F^2}$$

where $\hat{\mathbf{Y}} = \mathbf{K}_z \mathbf{A} \mathbf{K}_x$ and $\mathbf{A}$ is obtained from model estimates. Similarly, we are interested in the training MSE_y for analysing overfitting:

$$\text{Training MSE}_y = \frac{||\mathbf{M} \circ (\hat{\mathbf{Y}} - \mathbf{Y})||_F^2}{||\mathbf{M}||_F^2}.$$

B. Learning a Graph From Noisy Data

In order to evaluate the performance of the proposed model in learning a graph from noisy data, we add noise to the groundtruth data such that $\mathbf{Y} = \mathbf{K}_z \mathbf{A} \mathbf{K}_x + \mathbf{E}$, where every entry of the noise matrix $\mathbf{E}$ is an $i.i.d.$ sample from $\mathbf{E}_{ij} \sim \mathcal{N}(0, \sigma_\epsilon^2)$. We vary the noise level σ_ϵ^2 from 0 to 2 against which we plot the evaluation



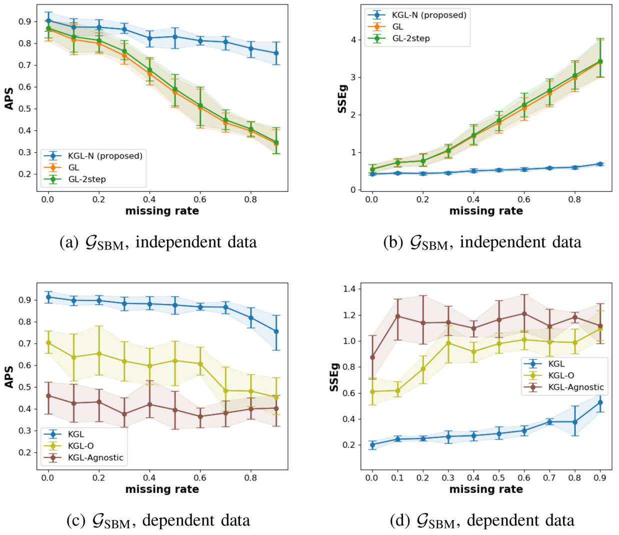
(a) $\mathcal{G}_{\mathrm{SBM}}$, independent data

(b) $\mathcal{G}_{\mathrm{SBM}}$, independent data

(c) $\mathcal{G}_{\mathrm{SBM}}$, dependent data

(d) $\mathcal{G}_{\mathrm{SBM}}$, dependent data

Fig. 2. The performance of recovering groundtruth graphs $\mathcal{G}_{\mathrm{SBM}}$ in terms of APS and $\mathrm{SSE}_{\mathcal{G}}$ from independent data (1st row) and dependent data (2nd row) with different noise levels. (a) $\mathcal{G}_{\mathrm{SBM}}$, independent data. (b) $\mathcal{G}_{\mathrm{SBM}}$, independent data. (c) $\mathcal{G}_{\mathrm{SBM}}$, dependent data. (d) $\mathcal{G}_{\mathrm{SBM}}$, dependent data.



(a) $\mathcal{G}_{\mathrm{SBM}}$, independent data

(b) $\mathcal{G}_{\mathrm{SBM}}$, independent data

(c) $\mathcal{G}_{\mathrm{SBM}}$, dependent data

(d) $\mathcal{G}_{\mathrm{SBM}}$, dependent data

Fig. 3. The performance of recovering groundtruth graphs $\mathcal{G}_{\mathrm{SBM}}$ in terms of APS and $\mathrm{SSE}_{\mathcal{G}}$ from independent data (1st row) and dependent data (2nd row) with different rates of missing values in $\mathbf{Y}$. (a) $\mathcal{G}_{\mathrm{SBM}}$, independent data. (b) $\mathcal{G}_{\mathrm{SBM}}$, independent data. (c) $\mathcal{G}_{\mathrm{SBM}}$, dependent data. (d) $\mathcal{G}_{\mathrm{SBM}}$, dependent data.

metrics in Fig. 2. Under the same settings of the noise level, random graph and model candidate, we repeat the experiment for 10 times and report the mean (the solid curves) as well as the 5th and 95th percentile (the error bars) of the evaluation metrics.

From Fig. 2, Fig. 8 and Fig. 9 (the latter two in Appendix E-A), the proposed models outperform the baseline models in terms of all evaluation metrics. Specifically, for $\mathcal{G}_{\mathrm{SBM}}$, when the data are independent, the performance of **KGL-N** drops slowly as the noise level increases, while that of **GL** and **GL-2step** drops quickly when noise level goes above 0.5. It is worth mentioning that the curves of two almost overlap in terms of APS. This indicates the identity mapping in **GL-2step** as denoising function does not help much in recovering graph structure, although it yields slightly smaller $\mathrm{SSE}_{\mathcal{G}}$ than **GL** with the noise level greater then 0.75.

For the dependent data, the proposed model **KGL** achieves a high performance when the noise level is less than 0.75. Even without the node-side information $\mathbf{K}_x$ as model input in **KGL** (note that the groundtruth data are generated in a consistent way in [45] proposing **KGL-agnostic**), the proposed method ($\mathbf{KGL-O}$) can still learn a meaningful graph with slightly worse performance compared to **KGL**. By contrast, **KGL-agnostic** cannot recover the groundtruth graph to a satisfying level even with little noise ($\sigma_\epsilon^2 = 0$), as its smoothness term does not capture the dependence structure on the observation side.

From Fig. 8 and Fig. 9, we see that $\mathcal{G}_{\mathrm{BA}}$ is slightly more difficult to recover from data, but an improvement can nevertheless be seen in the proposed models from the baselines when the noise level is low.

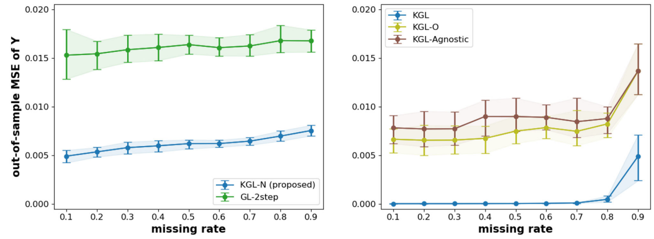
C. Learning a Graph From Missing Data

To examine the performance of learning a graph from incomplete data with **KGL** described in Section IV-E, we generate the mask matrix $\mathbf{M}$ indicating missing entries, where $\mathbf{M}_{ij} \overset{i.i.d.}{\sim} Bernoulli(1-r)$ and r is the missing rate, i.e. $\mathbf{M}_{ij}$ has a probability of r to be missing and has a value of 0. The preprocessed data with 0 replacing missing entries is $\mathbf{Y}_m = \mathbf{Y} \circ \mathbf{M}$, which is a natural choice in practice for the model candidates that cannot directly deal with missing entries, as the mean value of the entries in $\mathbf{Y}$ is 0 by design. We use $\mathbf{Y}_m$ as the input in the baseline models **GL** and **GL-2Step**. For **KGL-Agnostic** in Eq. (34), we also add a mask matrix $\mathbf{M}$ in the least-squares loss term to have a fair comparison.

We vary r from 0 to 0.9, against which we plot the evaluation metrics, APS and $\mathrm{SSE}_{\mathcal{G}}$, in Fig. 3. The plots for $\mathcal{G}_{\mathrm{ER}}$ and $\mathcal{G}_{\mathrm{BA}}$ are in Appendix E-B. Similar to the noisy scenario, we repeat the experiments 10 times under the same settings of missing rate, random graph and model candidate and report the mean (the solid curves) as well as the 5th and 95th percentile (the error bars) of the evaluation metrics.

For $\mathcal{G}_{\mathrm{SBM}}$, the proposed methods **KGL** and **KGL-N** can recover the groundtruth graphs reasonably well even when there are 80% missing entries. For the independent data scenario, the performance of the baseline models with the preprocessed data $\mathbf{Y}_m$ drops steeply as the missing rate increases. Although it can still recover the groundtruth graphs with a high APS and low $\mathrm{SSE}_{\mathcal{G}}$ when the missing rate is less than 20%, the performance is not as good as **KGL-N**. By contrast, for **KGL-N**, the APS only drops by 0.1 from no missing entries to around 90% missing entries, while $\mathrm{SSE}_{\mathcal{G}}$ only increases by around 0.05.

For the dependent data scenario, all three model candidates can deal with missing data directly by adding a mask matrix in the least-squares loss term. Consequently, their curves are relatively stable as the missing rate increases from 0 to 80%. However, the accuracy levels at which each of the model stabilises are different. The proposed method **KGL**, aware of both node-side and observation-side information, achieves the



(a) independent data

(b) dependent data

Fig. 4. The MSE of recovering missing entries in $\mathbf{Y}_m$. The results show the mean of 30 random experiments of 3 different types of graphs, i.e. 10 for each graph type. (a) independent data. (b) dependent data.

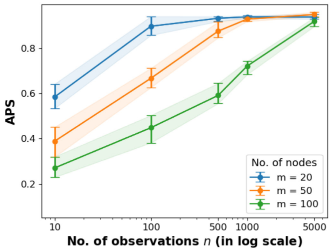


Fig. 5. The performance of learning graphs of different sizes m against varying number of observations n with the proposed model **KGL**. The results show the mean of 30 random experiments of 3 different types of graphs, i.e. 10 for each graph type.

highest APS and lowest $\text{SSE}_{\mathcal{G}}$. By contrast, $\mathbf{KGL-O}$, without access to the node-side information, can still recover a meaningful graph but with less correct edges and less accurate edge weights when the missing rate is less than 0.6. The performance of $\mathbf{KGL-Agnostic}$ is not as good as $\mathbf{KGL-O}$ because the smoothness term in Eq. (34) is not able to disentangle the influence of the observation-side dependency from the graph structure.

The performance of **KGL** does not differ much for different random graph models. Still, there is an improvement in recovering $\mathcal{G}_{\text{BA}}$ compared to the baseline graph learning models, as can be seen in Fig. 11(c) and Fig. 11(d) in Appendix E-B.

D. Graph-Structured Matrix Completion

In the experiment of learning a graph with incomplete data matrix in Section V-C, we are also interested in the performance of matrix completion. In Fig. 4, we plot the MSE_y of the missing entries (i.e. the out-of-sample MSE_y) that is averaged over all types of graphs against the varying missing rate. The proposed methods lead to much smaller errors compared to baseline models, for both independent and dependent data. It should be noted that **GL** does not offer a mechanism for inferring missing data, hence is not included in this experiment.

E. Learning a Graph of Different Sizes

The graph learning performance of the proposed method varies with the size of graphs and number of observations. As shown in Fig. 5, a hundred observations are sufficient to recover a small graph with $m = 20$ nodes with a high accuracy. When the graph size increases, the number of the observations required for **KGL** to achieve a high APS increases roughly exponentially. On the other hand, when the number of observations increases, the variance of APS of the learned graphs decreases.

F. Impact of Regularisation Hyperparameters

Three hyperparameters are involved in **KGL** and its variants. As introduced in Section IV, $\lambda > 0$ controls the complexity of the functional learning and prevents overfitting; $\rho > 0$ controls the relative importance of graph learning compared to functional learning, and at the same time determines the smoothness of the predicted data $\hat{\mathbf{y}}$ over $\mathbf{L} \otimes \mathbf{K}_z^{\dagger}$; Finally, $\psi > 0$ controls the Frobenius (ℓ_2) norm of the graph Laplacian which, together with the trace (ℓ_1) constraint, bears similarity to an elastic net regularisation [68]. The larger the ψ, the less sparse the graph with more uniform edge weights. The accuracy in learning the graph Laplacian $\mathbf{L}$ and inferring the data matrix $\mathbf{Y}$ is determined by the combination of these three hyperparameters, which is not straightforward to visualise and analyse at the same time. Fortunately, we may still gain some insights by examining their distinct effects separately.

Firstly, ψ should be chosen according to the prior belief of the graph sparsity defined as the number of edges with non-zero weights. As shown in Fig. 12 (in Appendix E-C), when $\psi \to 0$, the learned graph contains only a few most significant edge. Due to the constraint on the sum of edge weights, i.e. $\text{tr}(\mathbf{L}) = m$, the total weights m are allocated to a few significant edges when ψ is small. When $\psi \to \infty$, the learned graph becomes fully connected with equal edge weights. In the synthetic experiment where we have knowledge of the groundtruth graph, the best accuracy is obtained when the sparsity coincides with the groundtruth graph.

The value of ψ, on the other hand, has little effect on the accuracy of predicting the missing entries in $\mathbf{Y}$, as the update step of $\mathbf{A}$ does not involve the term with ψ. As shown in Fig. 13(a)-(d), the out-of-sample MSE_y is determined by the combination of λ and ρ. We first notice that the error is the same when $\lambda > 10^2$, showing that we overly penalise the function complexity in this case. Indeed, when $\lambda \to \infty$, the function is overly smooth such that the entries of the coefficient matrix $\mathbf{A}$ are all zero leading to the entries of prediction $\hat{\mathbf{Y}}$ being all zero as well. This also happens when $\rho \to \infty$, where the vector form of the prediction $\hat{\mathbf{y}}$ is forced to be overly smooth on $\mathbf{L} \otimes \mathbf{K}_z^{\dagger}$. In particular, when $\mathbf{K}_z^{\dagger} = \mathbf{I}_n$, every row vector of $\hat{\mathbf{Y}}$ (i.e. the predicted graph signal) has constant entries, as a result of minimising the term $\text{Tr}(\hat{\mathbf{Y}}^\top \mathbf{K}_z^{\dagger} \hat{\mathbf{Y}} \mathbf{L})$ to zero. On the other hand, when $\mathbf{K}_z^{\dagger} \neq \mathbf{I}_n$, all the entries in $\hat{\mathbf{Y}}$ has constant values such that this term is minimised to zero.

The ranges of values of λ and ρ for which the out-of-sample MSE_y is the smallest generally coincide with that for which the APS is the largest in Fig. 13 (in Appendix E-C), e.g. when $\psi = 10^{-5}, \lambda = 10^{-2}$ and $\rho = 10^{-2}$. However, the out-of-sample MSE_y is generally small when $\lambda < 0.1$ and $\rho < 0.1$. This is understandable as the function could be very complex with little penalisation, but this does not guarantee good performance in recovering the groundtruth graph.

VI. Real-World Experiments

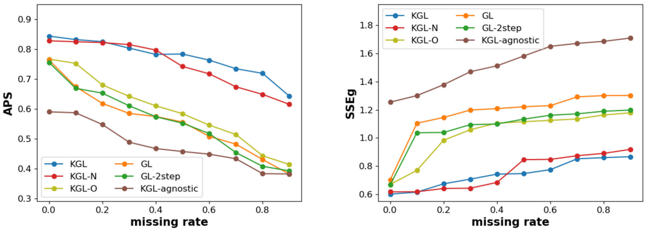
A. Swiss Temperature Data

In this experiment, we test the proposed models in learning a meteorological graph of 89 weather stations in Switzerland from the incomplete temperature data.[3] The raw data matrix contains 12 rows representing the temperatures of 12 months that are averaged over 30 years from 1981 to 2010 and 89 columns representing 89 measuring stations. The raw data are preprocessed such that each row has a zero mean.

To have a fair comparison, we deliberately omit a portion of the data as input for the model candidates and treat as groundtruth the learned graph obtained from $\mathbf{GL}$ using the complete 12-month data. Specifically, we only use the first three-month temperature records (i.e. the first three rows) to learn a graph. To test the performance in missing scenario, we further generate the mask matrix M with different rates of missing values, as described in Section V-C, and apply them to the three-month data. Similar to the synthetic experiment, we choose the hyperparameters in models that yield a graph density of 30% (i.e. keeping 30% most significant edges) to make the learned graphs comparable.

The altitude of a weather station is a useful node-side information for predicting temperature and learning a meteorological graph. Therefore, the corresponding kernel matrix $\mathbf{K}_x$ is obtained from an RBF kernel evaluated at the altitudes of each pair of weather stations for use in **KGL** and **KGL-N**. Monthly time-stamps correspond to the known observation-side information. The bandwidth parameter is chosen according to the median heuristic [67]. As described in Section V-A, $\mathbf{K}_z$ is thus obtained from an RBF kernel evaluated at three time-stamps for three-month graph signals and used as input in **KGL**, **KGL-Agnostic** and **KGL-O**. For **GL** and **GL-2step**, no side information is used. The hyperparameters are tuned via a grid search and highest performance achieved by the best set of hyperparameters is reported. For the real-world scenario where a groundtruth graph is not easy to obtain, the hyperparameter can be chosen according to the results in Section V-F.

We present the results in Fig. 6. In terms of both APS and SSE$_{\mathcal{G}}$, **KGL** and **KGL-N** outperform the other candidates. This indicates that altitude is a reliable node-side covariate with which we can learn a meaningful meteorological graph despite a small number of signals. When there are more missing values, **KGL**, with the known temporal information, slightly outperforms **KGL-N**. However, since we only have three-month signals, the temporal information is less predictive. Since the groundtruth graph is learned from $\mathbf{GL}$ with 12-month signals, the recovering ability of $\mathbf{GL}$ and $\mathbf{GL-2step}$ is not far behind when there is no missing values in three-month data, but drops sharply with an increasing missing rate. Compared to **KGL-O**, the poor performance of $\mathbf{KGL-Agnostic}$ indicates that the imprecise smoothness term in Eq. (34) is the main reason that we cannot recover an annual meteorological graph with only three-month temperature records, as both of them are agnostic to the node-side information.



(a) APS (b) SSE$_{\mathcal{G}}$

Fig. 6. The performance of learning a meteorological graph of 89 Swiss weather stations from the incomplete temperature data with various missing rates. (a) APS. (b) SSE$_{\mathcal{G}}$.

B. Sushi Review Data

In this experiment we will evaluate the performance of our proposed methods by comparing the recovered graphs with groundtruth using the Sushi review data collected in [69]. The authors tasked 5000 reviewers to rate 10 out of 100 sushis randomly with a score from 1 (least preferred) to 5 (most preferred); reviews for each sushi are treated as one graph signal in this experiment. For each reviewer, we have 10 descriptive features which cover demographical information about the reviewers, such as age, gender and the region the reviewer currently lives in. We also have 7 attributes describing each sushi, including its oiliness in taste, normalised price and its grouping (for example, red-meat fish sushi, white-meat fish sushi or shrimp sushi). We will treat the grouping attribute as the underlying groundtruth label for each sushi and not use it in the KGL algorithm.

We will consider 32 sushis from 5 sushi groups, namely red-meat (7 sushis), clam (6 sushis), blue-skinned fish (8 sushis), vegetable (6 sushis) and roe sushi (5 sushis). These are treated as the groundtruth labels. Our goal will be to recover a graph of sushis which contains clusters corresponding to these group labels (while omitting the group attribute from the node-side information, i.e. we retain only 6 remaining attributes).

We pick an increasing number of reviewers at random for our experiment. This is to demonstrate how the algorithm performs under different number of signals. After preprocessing, we arrive to a data matrix with 32 columns, each representing a type of sushi, and rows representing each reviewer's rating to the sushis. This is a sparse matrix with an average sparsity of 74%. We run **KGL**, **KGL-N**, **KGL-O**, **GL**, **GL-2step** and **KGL Agnostic** to obtain a graph of sushis. To evaluate the result quantitatively, we compute the *normalised mutual information* (NMI) between the cluster assignments obtained by applying spectral clustering [70] to the recovered graphs and the underlying sushi grouping. NMI is used to measure the agreement between two grouping assignments, 0 indicating no mutual information while 1 indicating perfect correlation. To emphasise that node-side

[3] The data are obtained from https://www.meteoswiss.admin.ch/home/climate/swiss-climate-in-detail/climate-normals/normal-values-per-measured-parameter.html

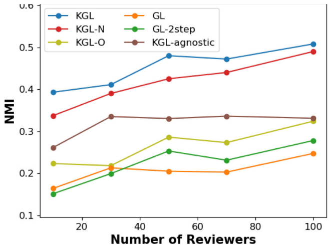


Fig. 7. Agreement between cluster assignments on learned sushi graphs and the withheld sushi group attributes. Graphs are learned from incomplete sushi review data.

attributes alone are not sufficient to recover the groundtruth label, we also applied clustering on the RBF graph obtained from remaining six sushi attributes, which resulted in an NMI score of 0.34.

Fig. 7 illustrated our results. **KGL** was the best performer, followed by **KGL-N**, and both significantly outperformed **KGL-Agnostic**, **KGL-O**, **GL-2step** and **GL**. Moreover, both **KGL** and **KGL-N** outperformed the case where we solely use the RBF Graph. The results demonstrate the merit of our proposed methods in incorporating side information for graph recovery, in particular the observation-side information (i.e. the reviewers' information) to capture the dependency between the observed signals.

VII. Conclusion

In this paper, we have revisited the smooth graph signals from a functional viewpoint and proposed a kernel-based graph learning framework that can integrate node-side and observation-side covariates. Specifically, we have designed a novel notion of smoothness of graph signals over the Kronecker product of two graph Laplacian matrices and combined it with a Kronecker product kernel regression of graph signals in order to capture the two-side dependency. We have shown the effectiveness and efficiency of the proposed method, via extensive synthetic and real-world experiments, demonstrating its usefulness in learning a meaningful topology from noisy, incomplete and dependent graph signals. Although we have proposed a fast implementation exploiting the Kronecker structure of kernel matrices, the computational complexity remains cubic in the maximum of the number of nodes and the number of signals. Hence, a natural future direction is to further reduce the computational complexity with the state-of-the-art methods for large-scale kernel-based learning, such as random Fourier features. Another interesting direction is to develop a generative graph learning model based upon the framework presented here, using connections between Gaussian processes and kernel methods.

Appendix A
Kronecker Product Kernel Regression

Taking a Bayesian viewpoint, we provide an interpretation of Eq. (7). From Eq. (8), maximising the log-posterior of the coefficient vector $\mathbf{a}$ leads to the objective in Eq. (7):

$$\begin{aligned}
&\max_{\mathbf{a}} \log p(\mathbf{a}|\mathbf{y}) \\
&= \max_{\mathbf{a}} \log p(\mathbf{y}|\mathbf{a}) + \log p(\mathbf{a}) \\
&= \max_{\mathbf{a}} -(\mathbf{y} - (\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a})^\top (\mathbf{y} - (\mathbf{K}_z \otimes \mathbf{K}_z\mathbf{a}) \\
&\qquad - \lambda\mathbf{a}^\top(\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a} \\
&= \min_{\mathbf{a}} ||\mathbf{y} - (\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a}||_2^2 + \lambda\mathbf{a}^\top(\mathbf{K}_x \otimes \mathbf{K}_z)\mathbf{a} \\
&= \min_{\mathbf{A}} ||\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x||_F^2 + \lambda \mathrm{Tr}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x\mathbf{A}^\top)
\end{aligned}$$

where λ is some constant parameter proportional to the variance of the noise σ_ϵ^2 in Eq. (8a).

Appendix B
Kronecker Product Laplacian-Like Operator

We define a Laplacian-like operator with a Kronecker product structure $\mathbf{L}_\otimes = \mathbf{L} \otimes \mathbf{L}_z$ for the data matrix with both node-side and observation-side dependency. Although $\mathbf{L}_\otimes$ may have positive off-diagonal entries and thus may not be a valid graph Laplacian matrix, it satisfies the following properties and a notion of frequency of $\mathbf{y}$ can be defined upon $\mathbf{L}_\otimes$:

- $\mathbf{L}_\otimes$ is symmetric and $\mathbf{L}_\otimes \cdot \mathbf{1} = \mathbf{0}$;
- $\mathbf{L}_\otimes$ admits the eigendecomposition

$$\begin{aligned}
\mathbf{L}_\otimes &= \mathbf{L} \otimes \mathbf{L}_z \\
&= (\mathbf{U} \otimes \mathbf{U}_z)(\mathbf{\Lambda} \otimes \mathbf{\Lambda}_z)(\mathbf{U} \otimes \mathbf{U}_z)^\top
\end{aligned}$$

where $\mathbf{U}$ and $\mathbf{\Lambda}$, and $\mathbf{U}_z$ and $\mathbf{\Lambda}_z$, are the eigenvector and eigenvalue matrices of the Laplacian matrices $\mathbf{L}$ and $\mathbf{L}_z$, respectively, and $\mathbf{U}_\otimes = (\mathbf{U} \otimes \mathbf{U}_z)$ is an orthogonal matrix and $\mathbf{\Lambda}_\otimes = (\mathbf{\Lambda} \otimes \mathbf{\Lambda}_z)$ is a diagonal matrix with real entries.

We can also obtain a two-dimensional graph Fourier transform $\check{\mathbf{Y}}$ of $\mathbf{Y}$ as in [71]:

$$\mathrm{vec}(\check{\mathbf{Y}}) = (\mathbf{U} \otimes \mathbf{U}_z)^\top \mathrm{vec}(\mathbf{Y}).$$

Appendix C
Proof of Positive Semi-Definiteness

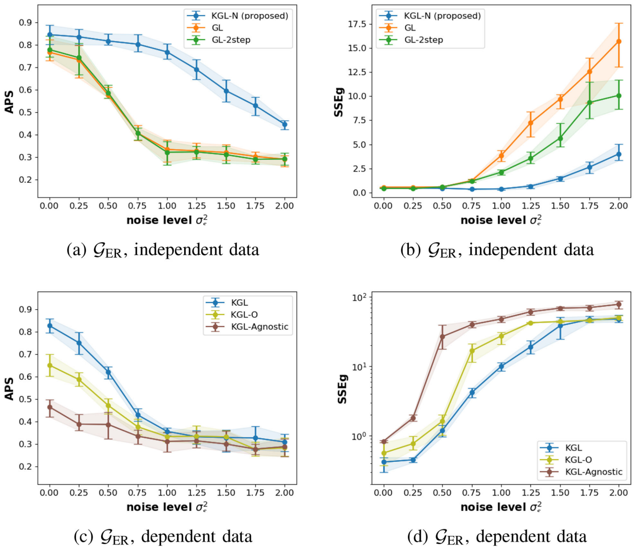
Lemma C.1: The matrix $\mathbf{C} = \mathbf{K}^2 + \lambda\mathbf{K} + \rho\mathbf{S} \otimes \mathbf{K}_z$ is positive semi-definite.

Proof: To prove $\mathbf{C}$ is positive semi-definite (p.s.d.), it suffices to show that the matrices $\mathbf{K}^2$, $\mathbf{K}$ and $\mathbf{S} \otimes \mathbf{K}_z$ are p.s.d.

As $\mathbf{K}_x$ and $\mathbf{K}_z$ are kernel matrices constructed by pairwise evaluations from two reproducing kernels $\kappa_\mathcal{X}$ and $\kappa_\mathcal{Z}$, they are p.s.d. The Kronecker product $\mathbf{K}$ is p.s.d, which is easy to prove from the eigendecomposition:

$$\begin{aligned}
\mathbf{K} &= \mathbf{K}_x \otimes \mathbf{K}_z \\
&= (\mathbf{U}_x\mathbf{\Lambda}_x\mathbf{U}_x^\top) \otimes (\mathbf{U}_z\mathbf{\Lambda}_z\mathbf{U}_z^\top) \qquad (35) \\
&= (\mathbf{U}_x \otimes \mathbf{U}_z)(\mathbf{\Lambda}_x \otimes \mathbf{\Lambda}_z)(\mathbf{U}_x \otimes \mathbf{U}_z)^\top
\end{aligned}$$

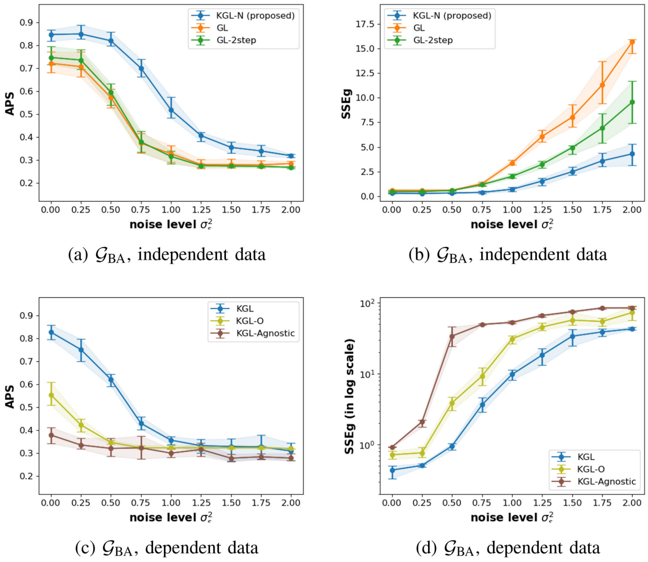
where $\mathbf{U} = (\mathbf{U}_x \otimes \mathbf{U}_z)$ is an orthogonal matrix and $\mathbf{\Lambda} = (\mathbf{\Lambda}_x \otimes \mathbf{\Lambda}_z)$ is a diagonal matrix with non-negative real entries.



(a) $\mathcal{G}_{\text{ER}}$, independent data (b) $\mathcal{G}_{\text{ER}}$, independent data

(c) $\mathcal{G}_{\text{ER}}$, dependent data (d) $\mathcal{G}_{\text{ER}}$, dependent data

Fig. 8. The performance of recovering groundtruth graphs $\mathcal{G}_{\text{ER}}$ from independent data (1st row) and dependent data (2nd row) with different noise levels. (a) $\mathcal{G}_{\text{ER}}$, independent data. (b) $\mathcal{G}_{\text{ER}}$, independent data. (c) $\mathcal{G}_{\text{ER}}$, dependent data. (d) $\mathcal{G}_{\text{ER}}$, dependent data.



(a) $\mathcal{G}_{\text{BA}}$, independent data (b) $\mathcal{G}_{\text{BA}}$, independent data

(c) $\mathcal{G}_{\text{BA}}$, dependent data (d) $\mathcal{G}_{\text{BA}}$, dependent data

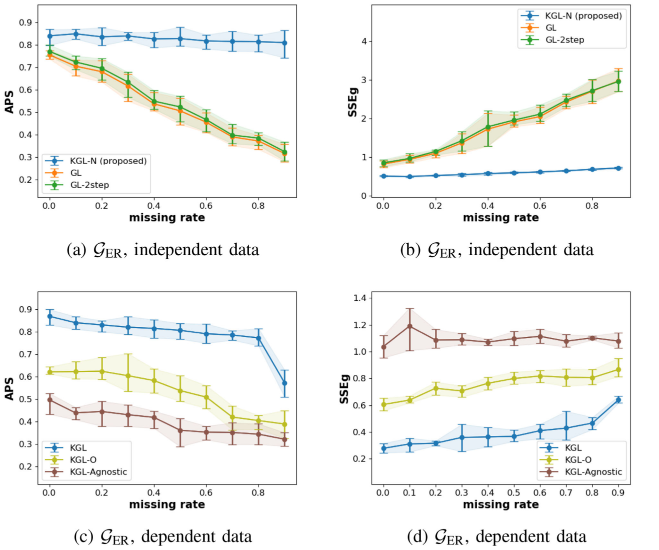
Fig. 9. The performance of recovering groundtruth graphs $\mathcal{G}_{\text{BA}}$ from independent data (1st row) and dependent data (2nd row) with different noise levels. (a) $\mathcal{G}_{\text{BA}}$, independent data. (b) $\mathcal{G}_{\text{BA}}$, independent data. (c) $\mathcal{G}_{\text{BA}}$, dependent data. (d) $\mathcal{G}_{\text{BA}}$, dependent data.

Next, $\mathbf{K}^2 = \mathbf{KK}$ is also p.s.d., as it is a product of commuting matrices and $\mathbf{K}^2$ preserves symmetry [72].

Finally, denote the column vectors of $\mathbf{K}_x$ as $[\mathbf{k}_1, \mathbf{k}_2, \ldots, \mathbf{k}_m]$, the weighted adjacency matrix as $\mathbf{W}$, and the non-negative edge weight between node j and j' as $w_{jj'}$. We have

$$\mathbf{S} = \mathbf{K}_x \mathbf{L} \mathbf{K}_x = \sum_{j \neq j'} w_{jj'} (\mathbf{k}_j - \mathbf{k}_{j'})(\mathbf{k}_j - \mathbf{k}_{j'})^\top$$

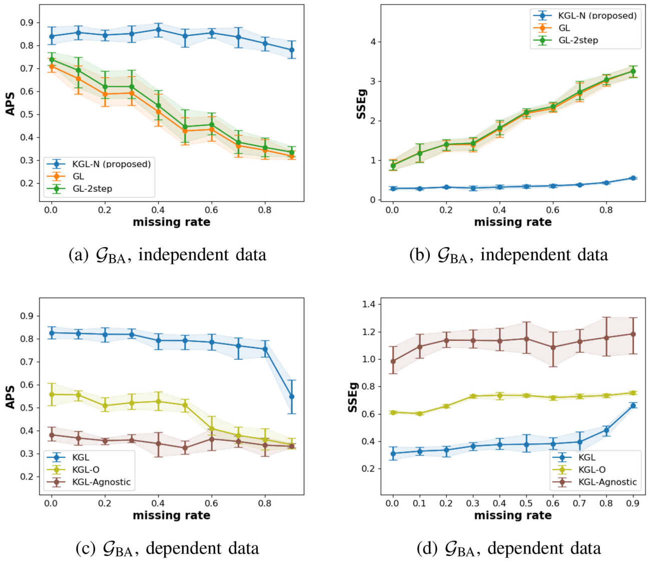
where $w_{jj'} \geq 0$. $\mathbf{S}$ can then be viewed as a weighted covariance matrix, which is symmetric and p.s.d. Therefore, $\mathbf{S} \otimes \mathbf{K}_z$ is p.s.d. following the same argument as for $\mathbf{K}$. ■



(a) $\mathcal{G}_{\text{ER}}$, independent data (b) $\mathcal{G}_{\text{ER}}$, independent data

(c) $\mathcal{G}_{\text{ER}}$, dependent data (d) $\mathcal{G}_{\text{ER}}$, dependent data

Fig. 10. The performance of recovering groundtruth graphs $\mathcal{G}_{\text{ER}}$ from independent data (1st row) and dependent data (2nd row) with different rates of missing values in $\mathbf{Y}$. (a) $\mathcal{G}_{\text{ER}}$, independent data. (b) $\mathcal{G}_{\text{ER}}$, independent data. (c) $\mathcal{G}_{\text{ER}}$, dependent data. (d) $\mathcal{G}_{\text{ER}}$, dependent data.



(a) $\mathcal{G}_{\text{BA}}$, independent data (b) $\mathcal{G}_{\text{BA}}$, independent data

(c) $\mathcal{G}_{\text{BA}}$, dependent data (d) $\mathcal{G}_{\text{BA}}$, dependent data

Fig. 11. The performance of recovering groundtruth graphs $\mathcal{G}_{\text{BA}}$ from independent data (1st row) and dependent data (2nd row) with different rates of missing values in $\mathbf{Y}$. (a) $\mathcal{G}_{\text{BA}}$, independent data. (b) $\mathcal{G}_{\text{BA}}$, independent data. (c) $\mathcal{G}_{\text{BA}}$, dependent data. (d) $\mathcal{G}_{\text{BA}}$, dependent data.

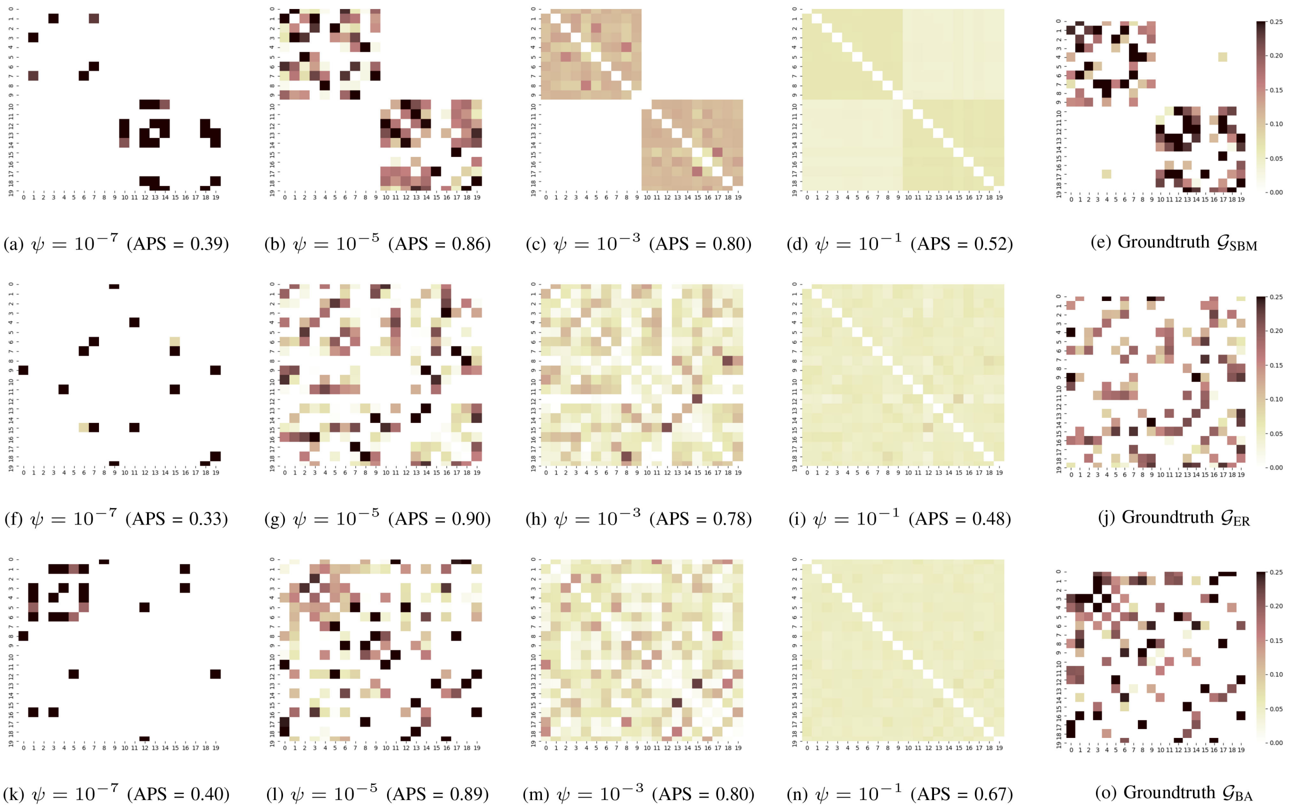
Appendix D
Derivation of Gradient in Eq. (33)

In this section, we show the derivation of Eq. (33), i.e. the gradient for updating $\mathbf{A}$ in the missing values scenario. Recall that the objective function is

$$J_{\mathbf{L}}(\mathbf{A}) = ||\mathbf{M} \circ (\mathbf{Y} - \mathbf{K}_z \mathbf{A} \mathbf{K}_x)||_F^2 + \lambda \mathrm{Tr}(\mathbf{K}_z \mathbf{A} \mathbf{K}_x \mathbf{A}^\top) + \rho \mathrm{Tr}(\mathbf{A} \mathbf{K}_x \mathbf{L} \mathbf{K}_x \mathbf{A}^\top \mathbf{K}_z).$$

With the following standard linear algebra identities for any matrices $\mathbf{D}, \mathbf{E}, \mathbf{F}$

- $||\mathbf{D} \circ \mathbf{E}||_F^2 = \mathrm{Tr}((\mathbf{D} \circ \mathbf{E})^\top (\mathbf{D} \circ \mathbf{E})) = \mathrm{Tr}(\mathbf{E}^\top (\mathbf{D} \circ \mathbf{E}))$



(a) $\psi = 10^{-7}$ (APS = 0.39) (b) $\psi = 10^{-5}$ (APS = 0.86) (c) $\psi = 10^{-3}$ (APS = 0.80) (d) $\psi = 10^{-1}$ (APS = 0.52) (e) Groundtruth $\mathcal{G}_{\mathrm{SBM}}$

(f) $\psi = 10^{-7}$ (APS = 0.33) (g) $\psi = 10^{-5}$ (APS = 0.90) (h) $\psi = 10^{-3}$ (APS = 0.78) (i) $\psi = 10^{-1}$ (APS = 0.48) (j) Groundtruth $\mathcal{G}_{\mathrm{ER}}$

(k) $\psi = 10^{-7}$ (APS = 0.40) (l) $\psi = 10^{-5}$ (APS = 0.89) (m) $\psi = 10^{-3}$ (APS = 0.80) (n) $\psi = 10^{-1}$ (APS = 0.67) (o) Groundtruth $\mathcal{G}_{\mathrm{BA}}$

Fig. 12. Graph sparsity with respect to ψ. The first row (a)-(d): the learned $\mathcal{G}_{\mathrm{SBM}}$; the second row (f)-(i): the learned $\mathcal{G}_{\mathrm{ER}}$; the third row (k)-(n): the learned $\mathcal{G}_{\mathrm{BA}}$, all from **KGL** with $\lambda = 10^{-1}$, $\rho = 10^{-2}$ and a fixed ψ. The respective groundtruth graphs are shown in the last column. (a) $\psi = 10^{-7}$ (APS = 0.39). (b) $\psi = 10^{-5}$ (APS = 0.86). (c) $\psi = 10^{-3}$ (APS = 0.80). (d) $\psi = 10^{-1}$ (APS = 0.52). (e) Groundtruth $\mathcal{G}_{\mathrm{SBM}}$. (f) $\psi = 10^{-7}$ (APS = 0.33). (g) $\psi = 10^{-5}$ (APS = 0.90). (h) $\psi = 10^{-3}$ (APS = 0.78). (i) $\psi = 10^{-1}$ (APS = 0.48). (j) Groundtruth $\mathcal{G}_{\mathrm{ER}}$. (k) $\psi = 10^{-7}$ (APS = 0.40). (l) $\psi = 10^{-5}$ (APS = 0.89). (m) $\psi = 10^{-3}$ (APS = 0.80). (n) $\psi = 10^{-1}$ (APS = 0.67). (o) Groundtruth $\mathcal{G}_{\mathrm{BA}}$.

- $\mathrm{vec}(\mathbf{D} \circ \mathbf{E}) = \mathrm{vec}(\mathbf{D}) \circ \mathrm{vec}(\mathbf{E})$
- $\mathrm{Tr}(\mathbf{D}^\top \mathbf{E}) = \mathrm{vec}(\mathbf{D})^\top \mathrm{vec}(\mathbf{E})$
- $\mathrm{vec}(\mathbf{DEF}) = (\mathbf{F}^\top \otimes \mathbf{D})\mathrm{vec}(\mathbf{E})$

the first term of $J_{\mathbf{L}}(\mathbf{A})$ becomes

$$\begin{aligned}&||\mathbf{M} \circ (\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x)||_F^2\\ &= \mathrm{Tr}\left((\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x)^\top (\mathbf{M} \circ (\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x))\right)\end{aligned}$$

By dropping constant terms, we have

$$\begin{aligned}&||\mathbf{M} \circ (\mathbf{Y} - \mathbf{K}_z\mathbf{A}\mathbf{K}_x)||_F^2\\ &= \mathrm{Tr}\left(-2(\mathbf{K}_z\mathbf{A}\mathbf{K}_x)^\top (\mathbf{M} \circ \mathbf{Y})\right)\\ &\quad + \mathrm{Tr}\left((\mathbf{K}_z\mathbf{A}\mathbf{K}_x)^\top (\mathbf{M} \circ (\mathbf{K}_z\mathbf{A}\mathbf{K}_x))\right)\\ &= -2\mathrm{vec}(\mathbf{A})^\top (\mathbf{K}_x \otimes \mathbf{K}_z)\mathrm{vec}(\mathbf{M} \circ \mathbf{Y})\\ &\quad + \mathrm{vec}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x)^\top (\mathrm{vec}(\mathbf{M}) \circ \mathrm{vec}(\mathbf{K}_z\mathbf{A}\mathbf{K}_x))\\ &= -2\mathbf{a}^\top \mathbf{K}\mathrm{vec}(\mathbf{M} \circ \mathbf{Y}) + \mathbf{a}^\top \mathbf{K}(\mathbf{m} \circ (\mathbf{K}\mathbf{a}))\end{aligned}$$

where $\mathbf{a} = \mathrm{vec}(\mathbf{A})$, $\mathbf{m} = \mathrm{vec}(\mathbf{M})$ and $\mathbf{K} = \mathbf{K}_x \otimes \mathbf{K}_z$. Putting it back to the objective and recognising the fact that $\mathbf{d} \circ \mathbf{e} = \mathrm{diag}(\mathbf{d})\mathbf{e}$ for two vectors $\mathbf{d}$ and $\mathbf{e}$, we have

$$\begin{aligned}J_{\mathbf{L}}(\mathbf{a}) =& -2\mathbf{a}^\top \mathbf{K}\mathrm{vec}(\mathbf{M} \circ \mathbf{Y}) + \mathbf{a}^\top \mathbf{K}(\mathbf{m} \circ (\mathbf{K}\mathbf{a}))\\ &+ \lambda\mathbf{a}^\top \mathbf{K}\mathbf{a} + \rho\mathbf{a}^\top (\mathbf{S} \otimes \mathbf{K}_z)\mathbf{a}\\ =& -2\mathbf{a}^\top \mathbf{K}\mathrm{vec}(\mathbf{M} \circ \mathbf{Y}) + \mathbf{a}^\top \mathbf{K}\mathrm{diag}(\mathbf{m})\mathbf{K}\mathbf{a}\\ &+ \lambda\mathbf{a}^\top \mathbf{K}\mathbf{a} + \rho\mathbf{a}^\top (\mathbf{S} \otimes \mathbf{K}_z)\mathbf{a}\end{aligned}$$

where $\mathbf{S} = \mathbf{K}_x\mathbf{L}\mathbf{K}_x$. We thus obtain the gradient for deriving Eq. (32) such that

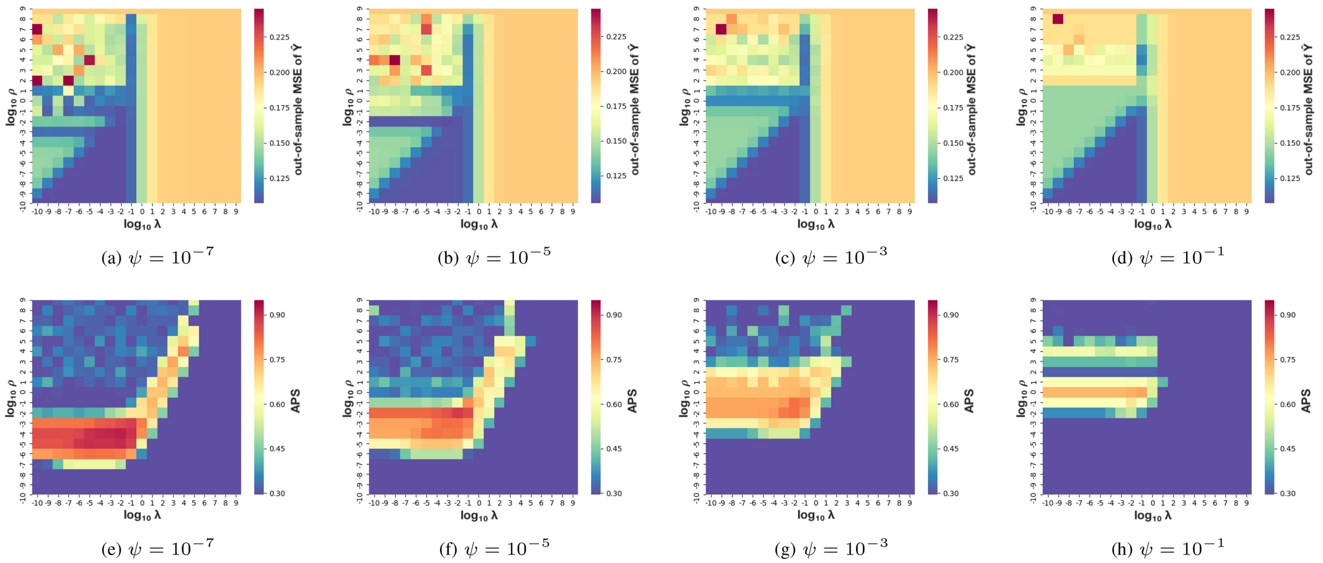
$$\nabla J_{\mathbf{L}}(\mathbf{a}) = -\mathbf{K}\mathrm{vec}(\mathbf{M} \circ \mathbf{Y}) + \mathbf{K}\mathrm{diag}(\mathbf{m})\mathbf{K}\mathbf{a} \tag{38a}$$

$$+ \lambda\mathbf{K}\mathbf{a} + \rho(\mathbf{S} \otimes \mathbf{K}_z)\mathbf{a}. \tag{38b}$$

Appendix E
Additional Results for Synthetic Experiments

A. Learning ER and BA Graphs From Noisy Data

Following the settings in Section V-B, we present in Fig. 8 and Fig. 9 the results of recovering $\mathcal{G}_{\mathrm{ER}}$ and $\mathcal{G}_{\mathrm{BA}}$ from independent data and dependent data with different noise levels, respectively.



(a) $\psi = 10^{-7}$ (b) $\psi = 10^{-5}$ (c) $\psi = 10^{-3}$ (d) $\psi = 10^{-1}$

(e) $\psi = 10^{-7}$ (f) $\psi = 10^{-5}$ (g) $\psi = 10^{-3}$ (h) $\psi = 10^{-1}$

Fig. 13. The out-of-sample MSE for data matrix $\mathbf{Y}$ (the first row) and the APS of the learned graph (the second row) with respect to λ and ρ, with 80% entries of $\mathbf{Y}$ as training sample from **KGL** with a fixed ψ. (a) $\psi = 10^{-7}$. (b) $\psi = 10^{-5}$. (c) $\psi = 10^{-3}$. (d) $\psi = 10^{-1}$. (e) $\psi = 10^{-7}$. (f) $\psi = 10^{-5}$. (g) $\psi = 10^{-3}$. (h) $\psi = 10^{-1}$.

B. Learning ER and BA Graphs From Missing Data

Following the settings in Section V-B, we present in Fig. 10 and Fig. 11 the results of recovering $\mathcal{G}_{\text{ER}}$ and $\mathcal{G}_{\text{BA}}$ from independent data and dependent data with different missing rates, respectively.

C. Impact of Regularisation Hyperparameters

Fig. 12 and Fig. 13 illustrate the learning performance with respect to the three hyperparameters in the proposed **KGL** model in Section V-F.

References

[1] J. Berant, A. Chou, R. Frostig, and P. Liang, "Semantic parsing on freebase from question-answer pairs," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2013, pp. 1533–1544.

[2] A. Andoni and P. Indyk, "Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions," in *Proc. 47th Annu. IEEE Symp. Foundations Comput. Sci.*, 2006, pp. 459–468.

[3] M. Muja and D. G. Lowe, "Fast approximate nearest neighbors with automatic algorithm configuration," in *Proc. Int. Conf. Comput. Vis. Theory Appl.*, vol. 1, 2009, pp. 331–340.

[4] J. L. Bentley, "Multidimensional binary search trees used for associative searching," *Commun. ACM*, vol. 18, no. 9, pp. 509–517, 1975.

[5] W. Dong, C. Moses, and K. Li, "Efficient k-nearest neighbor graph construction for generic similarity measures," in *Proc. 20th Int. Conf. World Wide Web*, 2011, pp. 577–586.

[6] S. L. Lauritzen, *Graphical Models*. Oxford, U.K.: Clarendon Press, 1996.

[7] D. Shuman, S. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Process. Mag.*, vol. 3, no. 30, pp. 83–98, May 2013.

[8] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE Proc. IRE*, vol. 106, no. 5, pp. 808–828, May 2018.

[9] A. Sandryhaila and J. M. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, no. 7, pp. 1644–1656, Apr. 2013.

[10] G. Mateos, S. Segarra, A. G. Marques, and A. Ribeiro, "Connecting the dots: Identifying network structure via graph signal processing," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 16–43, May 2019.

[11] X. Dong, D. Thanou, M. Rabbat, and P. Frossard, "Learning graphs from data: A signal representative perspective," *IEEE Signal Process. Mag.*, vol. 36, no. 3, pp. 44–63, May 2019.

[12] D. Thanou, X. Dong, D. Kressner, and P. Frossard, "Learning heat diffusion graphs," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 484–499, Sep. 2017.

[13] B. Pasdeloup, V. Gripon, G. Mercier, D. Pastor, and M. G. Rabbat, "Characterization and inference of graph diffusion processes from observations of stationary signals," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 4, no. 3, pp. 481–496, Sep. 2018.

[14] R. Shafipour, S. Segarra, A. G. Marques, and G. Mateos, "Identifying the topology of undirected networks from diffused non-stationary graph signals," 2019, *arXiv:1801.03862v2*.

[15] S. Segarra, A. G. Marques, G. Mateos, and A. Ribeiro, "Network topology inference from spectral templates," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 3, pp. 467–483, Sep. 2017.

[16] V. Kalofolias, "How to learn a graph from smooth signals," in *Proc. Artif. Intell. Statist.*, 2016, pp. 920–929.

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

[18] V. Kalofolias and N. Perraudin, "Large scale graph learning from smooth signals," in *Proc. Int. Conf. Learn. Representations*, 2018.

[19] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under laplacian and structural constraints," *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 6, pp. 825–841, Sep. 2017.

[20] S. P. Chepuri, S. Liu, G. Leus, and A. O. Hero, "Learning sparse graphs under smoothness prior," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 6508–6512.

[21] P. Berger, G. Hannak, and G. Matz, "Efficient graph learning from noisy and incomplete data," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 105–119, 2020.

[22] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, "Neural relational inference for interacting systems," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 2688–2697.

[23] L. Franceschi, M. Niepert, M. Pontil, and X. He, "Learning discrete structures for graph neural networks," in *Proc. 36th Int. Conf. Mach. Learn.*, vol. 97. PMLR, 2019, pp. 1972–1982.

[24] W. Xiong, T. Hoang, and W. Y. Wang, "Deeppath: A. reinforcement learning method for knowledge graph reasoning," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2017, pp. 564–573.

[25] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," in *Proc. Int. Conf. Learning Representations*, 2017.

[26] O. Banerjee, L. El Ghaoui, and A. D'Aspremont, "Model selection through sparse maximum likelihood estimation for multivariate gaussian or binary data," *J. Mach. Learn. Res.*, vol. 9, pp. 485–516, 2008.

[27] A. D'Aspremont, O. Banerjee, and L. El Ghaoui, "First-order methods for sparse covariance selection," *SIAM J. Matrix Anal. Appl.*, vol. 30, no. 1, pp. 56–66, 2008.

[28] R. Mazumder and T. Hastie, "The graphical lasso: New insights and alternatives," *Electron. J. Statist.*, vol. 6, no. Aug., pp. 2125–2149, 2012.

[29] C.-J. Hsieh, I. S. Dhillon, P. K. Ravikumar, and M. A. Sustik, "Sparse inverse covariance matrix estimation using quadratic approximation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2330–2338.

[30] Z. Lu, "Adaptive first-order methods for general sparse inverse covariance selection," *SIAM J. Matrix Anal. Appl.*, vol. 31, no. 4, pp. 2000–2016, 2009.

[31] J. Friedman, T. Hastie, and R. Tibshirani, "Sparse inverse covariance estimation with the graphical lasso," *Biostatistics*, vol. 9, no. 3, pp. 432–441, 2008.

[32] B. Rolfs, B. Rajaratnam, D. Guillot, I. Wong, and A. Maleki, "Iterative thresholding algorithm for sparse inverse covariance estimation," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1574–1582.

[33] C.-J. Hsieh, M. A. Sustik, I. S. Dhillon, P. K. Ravikumar, and R. Poldrack, "Big & quic: Sparse inverse covariance estimation for a million variables," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3165–3173.

[34] B. Lake and J. Tenenbaum, "Discovering structure by learning sparse graphs," in *Proc. Annu. Meeting Cogn. Sci. Soc.*, vol. 32, no. 32, 2010.

[35] M. Slawski and M. Hein, "Estimation of positive definite m-matrices and structure learning for attractive gaussian markov random fields," *Linear Algebra Its Appl.*, vol. 473, pp. 145–179, 2015.

[36] H. E. Egilmez, E. Pavez, and A. Ortega, "Graph learning from data under laplacian and structural constraints," *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 6, pp. 825–841, Sep. 2017.

[37] Z. Deng and A. M.-C. So, "A fast proximal point algorithm for generalized graph laplacian learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 5425–5429.

[38] O. Stegle, C. Lippert, J. M. Mooij, N. D. Lawrence, and K. Borgwardt, "Efficient inference in matrix-variate gaussian models with iid observation noise," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 630–638.

[39] K. Greenewald, S. Zhou, and A. Hero III, "Tensor graphical lasso (teralasso)," *J. Roy. Stat. Soc.: Ser. B Stat. Methodol.*, vol. 81, no. 5, pp. 901–931, 2019.

[40] X. Zheng, B. Aragam, P. K. Ravikumar, and E. P. Xing, "Dags with no tears: Continuous optimization for structure learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 9472–9483.

[41] G. B. Giannakis, Y. Shen, and G. V. Karanikolas, "Topology identification and learning over graphs: Accounting for nonlinearities and dynamics," *Proc. IEEE Proc. IRE*, vol. 106, no. 5, pp. 787–807, May 2018.

[42] R. Pamfil *et al.*, "Dynotears: Structure learning from time-series data," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 1595–1605.

[43] V. N. Ioannidis, Y. Shen, and G. B. Giannakis, "Semi-blind inference of topologies and signals over graphs," in *Proc. IEEE Data Sci. Workshop*, 2018, pp. 165–169.

[44] V. N. Ioannidis, Y. Shen, and G. B. Giannakis, "Semi-blind inference of topologies and dynamical processes over dynamic graphs," *IEEE Trans. Signal Process.*, vol. 67, no. 9, pp. 2263–2274, May 2019.

[45] A. Venkitaraman, S. Chatterjee, and P. Händel, "Predicting graph signals using kernel regression where the input signal is agnostic to a graph," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 5, no. 4, pp. 698–710, Dec. 2019.

[46] Y. Liu, L. Yang, K. You, W. Guo, and W. Wang, "Graph learning based on spatiotemporal smoothness for time-varying graph signal," *IEEE Access*, vol. 7, pp. 62 372–62 386, 2019.

[47] Y. Liu, W. Guo, K. You, L. Zhao, T. Peng, and W. Wang, "Graph learning for spatiotemporal signal with long-and short-term characterization," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 699–713, 2020.

[48] V. Kalofolias, A. Loukas, D. Thanou, and P. Frossard, "Learning time varying graphs," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2017, pp. 2826–2830.

[49] K. Yamada, Y. Tanaka, and A. Ortega, "Time-varying graph learning based on sparseness of temporal variation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2019, pp. 5411–5415.

[50] Z. Zhang, P. Cui, and W. Zhu, "Deep learning on graphs: A survey," *IEEE Trans. Knowl. Data Eng.*, to be published, doi: 10.1109/TKDE.2020.2981333.

[51] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 1, pp. 4–24, Jan. 2021.

[52] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst, "Geometric deep learning: Going beyond euclidean data," *IEEE Signal Process. Mag.*, vol. 34, no. 4, pp. 18–42, Jul. 2017.

[53] V. Hadziosmanovic, Y. Li, X. Liu, S. Kim, D. Dynerman, and L. Royer, "Graph learning networks," in *Proc. Int. Conf. Mach. Learn. Work. Learn. Reason. With Graph-Structured Represent*, 2019.

[54] Z. Wu, S. Pan, G. Long, J. Jiang, X. Chang, and C. Zhang, "Connecting the dots: Multivariate time series forecasting with graph neural networks," 2020, *arXiv:2005.11650*.

[55] B. Jiang, Z. Zhang, D. Lin, J. Tang, and B. Luo, "Semi-supervised learning with graph learning-convolutional networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 11 313–11 320.

[56] L. Yang, Z. Kang, X. Cao, D. Jin, B. Yang, and Y. Guo, "Topology optimization based graph convolutional network," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 4054–4061.

[57] Y. Chen, L. Wu, and M. J. Zaki, "Deep iterative and adaptive learning for graph neural networks," 2019, *arXiv:1912.07832*.

[58] Y. Saatçi, "Scalable inference for structured gaussian process models," Ph.D. dissertation, Citeseer.

[59] S. Ding, R. D. Cook, "Matrix variate regressions and envelope models," *J. Roy. Stat. Soc. Ser. B*, vol. 80, no. 2, pp. 387–408, 2018.

[60] A. J. Smola and R. Kondor, "Kernels and regularization on graphs," *Learning Theory and Kernel Machines*. Berlin, Germany: Springer, 2003, pp. 144–158.

[61] M. Belkin, P. Niyogi, and V. Sindhwani, "Manifold regularization: A. geometric framework for learning from labeled and unlabeled examples," *J. Mach. Learn. Res.*, vol. 7, no. Nov, pp. 2399–2434, 2006.

[62] P. Tseng, "Convergence of a block coordinate descent method for nondifferentiable minimization," *J. Optim. Theory Appl.*, vol. 109, no. 3, pp. 475–494, 2001.

[63] J. Gorski, F. Pfeuffer, and K. Klamroth, "Biconvex sets and optimization with biconvex functions: A. survey and extensions," *Math. Methods Oper. Res.*, vol. 66, no. 3, pp. 373–407, 2007.

[64] S. Diamond and S. Boyd, "CVXPY: A python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.

[65] S. Kumar, M. Mohri, and A. Talwalkar, "Sampling methods for the nyström method," *The J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 981–1006, 2012.

[66] D. P. Francis and K. Raimond, "Major advancements in kernel function approximation," *Artif. Intell. Rev.*, 2020.

[67] D. Garreau, W. Jitkrittum, and M. Kanagawa, "Large sample analysis of the median heuristic," 2017, *arXiv:1707.07269*.

[68] H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *J. Roy. Stat. Soc.: Ser. B. Stat. Methodol.*, vol. 67, no. 2, pp. 301–320, 2005.

[69] T. Kamishima, "Nantonac collaborative filtering: Recommendation based on order responses," in *Proc. 9th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2003, pp. 583–588.

[70] U. V. Luxburg, "A tutorial on spectral clustering," *Statist. Comput.*, vol. 17, no. 4, pp. 395–416, 2007.

[71] F. Monti, M. Bronstein, and X. Bresson, "Geometric matrix completion with recurrent multi-graph neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 3697–3707.

[72] A. Meenakshi and C. Rajian, "On a product of positive semidefinite matrices," *Linear Algebra Its Appl.*, vol. 295, no. 1-3, pp. 3–6, 1999.