Product Graph Gaussian Processes for Multi-domain Data Imputation and Active Learning

Sai Kiran Kadambari and Sundeep Prabhakar Chepuri
Indian Institute of Science, Bengaluru, India

***Abstract*—In this work, we consider the problem of imputing signals defined on the nodes of a product graph from the subset of observations. To this end, we focus on learning their predictive probability distribution function (PDF) based Gaussian processes. In particular, we propose a product graph Gaussian process model, which incorporates the product graph structure in the Gaussian process kernel via a product graph filter. When the observed graph signals are real-valued, the mean and variance of the predictive PDF can be computed in closed form. Further, the variance captures the model uncertainty, which we use for active learning to obtain subsequent observations. We demonstrate the efficacy of the proposed method on multi-domain data imputation and active learning tasks on synthetic and real-world datasets.**

***Index Terms*—Gaussian process, graphs, product graphs, semi-supervised learning over graphs.**

I. Introduction

In many applications, we observe large volumes of data or signals supported on non-Euclidean (the irregular) domains or networks. Such data can be modeled as the signals indexed by the nodes of a graph (hence referred to as graph signals) and the pairwise interaction between the data points are modeled as edges [1]. Often, graphs underlying data are factorizable and can be expressed as a product of two or more smaller graphs [2], [3]. Such factorizable graphs are called *product graphs*, which can represent multi-domain graph data. For example, in time-series data collected from a sensor network, there is a spatial and a temporal domain. Each space-time measurement is then enumerated by the nodes of a product graph, e.g., formed by the Cartesian product of a spatial domain graph and a temporal domain graph. Computations involving smaller graph factors have well-documented merits for various semi-supervised learning tasks like regression, classification and matrix completion, to name a few.

A common graph machine learning task is to interpolate or predict graph signals on unobserved nodes from observations on a subset of nodes [4] . Ignoring the product graph structure, recent works [4], [5] develop point estimators for imputing missing graph signals, assuming that the underlying graph signals are smooth on the graph. However, these methods do not quantify the uncertainty or confidence in the estimate, which can be useful for active learning (AL). Gaussian process (GP) methods [6] follow a Bayesian approach to model the uncertainty in the estimate by learning its predictive probability distribution function (PDF). For the graph signals, graph Gaussian process (GGP) leverage the underlying graph structure to estimate a function predictive PDF of the graph signals on the unobserved nodes. These methods incorporate graph information through a graph kernel or graph filter [7], [8]. These methods typically incur a cubic complexity (cubic in the number of nodes) (as discussed later on in Section III) and, thus, are not scalable for large-scale multi-domain graph data. For the Euclidean domain data, scalable GP methods for regression tasks are proposed [9], [10]. These methods exploit the fact that the kernel underlying such data can be factorized as the Kronecker product of kernels related to each domain. Even though these methods are scalable, they are limited to regular domain datasets.

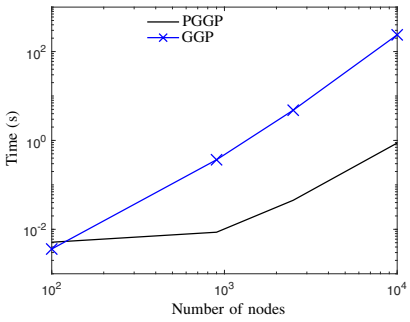

Fig. 1: Average run time of PGGP (proposed method) and GGP [8].

In this work, given a multi-domain graph signal observations on a subset of nodes of a product graph, we focus on learning a function predictive posterior PDF corresponding to the signals on the remaining nodes along with the model uncertainty. Specifically, the contributions of this paper are as follows. Following a GP approach, we propose a product graph Gaussian process (PGGP). We incorporate the underlying product graph structure into the GP kernel via a generalized product graph filter with learnable parameters. Assuming the observed graph signals are real-valued and follow a Kronecker-structured sampling scheme (as in [11]), the function predictive posterior PDF of the graph signals related to the unobserved nodes follows Gaussian distribution and its mean and variance can be computed in closed form. We leverage the model uncertainty captured by the variance for active learning to select subsequent observations. We demonstrate proposed method on synthetic and real-world datasets on graph data interpolation and active learning tasks. The proposed method is scalable and is particularly useful for processing signals on large-scale product graphs (see the average run time of GGP and PGGP in Fig. 1).

Throughout the paper, we denote matrices (respectively, vectors) with boldface uppercase (lower case) letters and sets with calligraphic letters. Given a square matrix $\mathbf{A}$, the trace

and determinant of a matrix $\mathbf{A}$ are represented by $\mathrm{tr}(\mathbf{A})$ and $\det(\mathbf{A})$, respectively. For any three matrices $\mathbf{A}$, $\mathbf{B}$, $\mathbf{C}$ of appropriate dimension, the following holds

$$\mathrm{vec}\left(\mathbf{ABC}\right) = \mathrm{vec}\left(\mathbf{C}^T \otimes \mathbf{A}\right) \mathrm{vec}\left(\mathbf{B}\right). \tag{1}$$

II. Background

Consider an undirected graph $\mathcal{G}_N = (\mathcal{V}_N, \mathcal{E}_N)$ with N nodes whose vertex set is $\mathcal{V}_N$ and the edge set $\mathcal{E}_N$. We represent the structure of the graph by an adjacency matrix $\mathbf{A}_N \in \mathbb{R}^{N\times N}$. When there is an edge between the nodes i and j, $[\mathbf{A}_N]_{i,j} \neq 0$ and $[\mathbf{A}_N]_{i,j} = 0$, otherwise. We define a corresponding normalized graph Laplacian matrix $\mathbf{L}_N = \mathbf{I}_N - \mathbf{D}_N^{-\frac{1}{2}}\mathbf{A}_N\mathbf{D}_N^{-\frac{1}{2}}$, where $\mathbf{D}_N = \mathrm{diag}(\mathbf{A}_N\mathbf{1}) \in \mathbb{R}^{\mathbf{N}\times\mathbf{N}}$ is the diagonal degree matrix. Let us collect the D_N dimensional features related the nodes of the graph $\mathcal{G}_N$ in the feature matrix $\mathbf{\Theta}_N \in \mathbb{R}^{D_N\times N}$ and the graph signal in a vector $\mathbf{y} \in \mathbb{R}^N$. Given a normalized graph Laplacian matrix $\mathbf{L}_N$, we define a k_Nth order finite impulse response (FIR) graph filter as $\mathbf{H}_N = \sum_{i=0}^{k_N} \alpha_i \mathbf{L}_N^i$, where $\boldsymbol{\alpha}_N = [\alpha_0, \alpha_1, \ldots, \alpha_N]^T \in \mathbb{R}^{k_N+1}$ are the filter coefficients.

Assume that $\mathcal{G}_N$ can be factorized as a product of the two graphs $\mathcal{G}_P = (\mathcal{V}_P, \mathcal{E}_P)$ and $\mathcal{G}_Q = (\mathcal{V}_Q, \mathcal{E}_Q)$ with P and Q nodes. Let their corresponding normalized graph Laplacian matrices be the $\mathbf{L}_P \in \mathbb{R}^{P\times P}$ and $\mathbf{L}_Q \in \mathbb{R}^{Q\times Q}$, respectively. The vertex set of the product graph $\mathcal{G}_N$ is $\mathcal{V}_N = \mathcal{V}_P \times \mathcal{V}_Q$ with $N = PQ$ and the edge set of $\mathcal{G}_N$ depends on the type of product graph [2]. We call $\mathcal{G}_P$ and $\mathcal{G}_Q$ as the factors of the product graph. Let us collect the D_P (respectively, D_Q) dimensional features on the graph factor in the matrices $\mathbf{\Theta}_P \in \mathbb{R}^{D_P\times P}$ ($\mathbf{\Theta}_Q \in \mathbb{R}^{D_Q\times Q}$). Consider a signal $\mathbf{y} \in \mathbb{R}^N$ defined on the nodes of the product graph. As the each node in the product graph is indexed by a pair of nodes in its graph factors, we have $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$, where $\mathbf{Y} \in \mathbb{R}^{P\times Q}$.

Given the normalized graph Laplacian matrices $\mathbf{L}_P$ and $\mathbf{L}_Q$ of the product graph factors, we define a product graph FIR filter as

$$\mathbf{H}(\mathbf{L}_P, \mathbf{L}_Q) = \sum_{k=0}^{k_P}\sum_{l=0}^{k_Q} h_{kl}\left(\mathbf{L}_P^k \otimes \mathbf{L}_Q^l\right) \tag{2}$$

where $\{h_{kl}\}$ are the filter coefficients and k_P and k_Q are the filter orders. Assuming $h_{kl} = \alpha_k\beta_l$ and using the properties of the Kronecker product, we re-write the product graph filter as

$$\mathbf{H}(\mathbf{L}_P, \mathbf{L}_Q) = \mathbf{H}_Q^T \otimes \mathbf{H}_P, \tag{3}$$

where $\mathbf{H}_P = \sum_{k=0}^{k_P}\alpha_k\mathbf{L}_P^k$ and $\mathbf{H}_Q = \sum_{l=0}^{k_Q}\beta_l\mathbf{L}_Q^k$ are the FIR graph filters of order k_P and k_Q defined on graph factors $\mathcal{G}_P$ and $\mathcal{G}_Q$, respectively. Here, $\boldsymbol{\alpha} = \left[\alpha_0, \alpha_1, \ldots, \alpha_{k_p}\right]^T \in \mathbb{R}^{k_p+1}$ and $\boldsymbol{\beta} = \left[\beta_0, \beta_1, \ldots, \beta_{k_q}\right]^T \in \mathbb{R}^{k_q+1}$ are the filter coefficients. Given a product graph signal $\mathbf{y}$, the filtered graph signal $\tilde{\mathbf{y}} \in \mathbb{R}^N$ is $\tilde{\mathbf{y}} = \left(\mathbf{H}_Q^T \otimes \mathbf{H}_P\right)\mathbf{y}$. Upon reshaping $\tilde{\mathbf{y}}$ as $\tilde{\mathbf{Y}} \in \mathbb{R}^{P\times Q}$, we have

$$\tilde{\mathbf{Y}} = \mathbf{H}_Q\mathbf{Y}\mathbf{H}_P. \tag{4}$$

This means that when the product graph filter is factorizable, filtering a product graph signal is equivalent to filtering the columns and rows of $\mathbf{Y}$ with the graph filters $\mathbf{H}_P$ and $\mathbf{H}_Q$, respectively.

III. Product graph Gaussian process

In this section, we state the data imputation problem on product graphs and develop a computationally efficient solution leveraging the properties of product graphs and the Kronecker product of matrices.

Given noisy graph signals on a subset of nodes of a product graph, we are interested in estimating the signals on the remaining nodes as a function predictive PDF and obtain the model uncertainty. In this work, we consider a structured sampler of product graph signals [11], where selecting a subset of nodes on the product graph $\mathcal{G}_N$ is equivalent to selecting a subset of nodes from its graph factors $\mathcal{G}_P$ and $\mathcal{G}_Q$, respectively. Then, the subsampled product graph signals from the set of observed (respectively, unobserved) nodes collected in the matrices $\mathbf{Y}_\mathrm{o} \in \mathbb{R}^{P_\mathrm{o}\times Q_\mathrm{o}}$ (respectively $\mathbf{Y}_\mathrm{u} \in \mathbb{R}^{P_\mathrm{u}\times Q_\mathrm{u}}$) are given by

$$\mathbf{Y}_\mathrm{o} = \mathbf{S}_P\mathbf{Y}\mathbf{S}_Q^T, \text{ and } \mathbf{Y}_\mathrm{u} = \bar{\mathbf{S}}_P\mathbf{Y}\bar{\mathbf{S}}_Q^T. \tag{5}$$

The matrices $\mathbf{S}_P \in \{0,1\}^{P_\mathrm{o}\times P}$ and $\mathbf{S}_Q \in \{0,1\}^{Q_\mathrm{o}\times Q}$ (respectively, $\bar{\mathbf{S}}_P \in \{0,1\}^{P_\mathrm{u}\times P}$ and $\bar{\mathbf{S}}_Q \in \{0,1\}^{Q_\mathrm{u}\times Q}$) are the sampling matrices that select graph signals on the subset of nodes in each graph factor $\mathcal{G}_P$ and $\mathcal{G}_Q$.

We assume the observed product graph signals are noisy. Let the observed noisy product graph signal be $\mathbf{X}_\mathrm{o} \in \mathbb{R}^{P_\mathrm{o}\times Q_\mathrm{o}}$ given by $\mathbf{X}_\mathrm{o} = \mathbf{S}_P\mathbf{Y}\mathbf{S}_Q^T + \mathbf{E}$, where $\mathbf{Y} \in \mathbb{R}^{P\times Q}$ is the true graph signal and $\mathbf{E} \in \mathbb{R}^{P_\mathrm{o}\times Q_\mathrm{o}}$ is the noise matrix with $[\mathbf{E}]_{i,j} = \mathcal{N}(0, \sigma^2)$. Given the noisy product graph signal $\mathbf{X}_\mathrm{o} \in \mathbb{R}^{P_\mathrm{o}\times Q_\mathrm{o}}$ and the graph Laplacian matrices of the graph factors, we aim to estimate the posterior predictive PDF of the product graph signals $\mathbf{Y}_\mathrm{u} \in \mathbb{R}^{P_\mathrm{u}\times Q_\mathrm{u}}$. Upon vectorization, using (1), the matrices $\mathbf{X}_\mathrm{o}$ and $\mathbf{Y}_\mathrm{u}$ can be equivalently represented as

$$\mathbf{x}_\mathrm{o} = \left(\mathbf{S}_Q \otimes \mathbf{S}_P\right)\mathbf{y} + \mathbf{e}, \text{ and } \mathbf{y}_\mathrm{u} = \left(\bar{\mathbf{S}}_Q \otimes \bar{\mathbf{S}}_Q\right)\mathbf{y}, \tag{6}$$

where $\mathbf{x}_\mathrm{o} = \mathrm{vec}(\mathbf{X}_\mathrm{o})$, $\mathbf{y}_\mathrm{o} = \mathrm{vec}(\mathbf{Y}_\mathrm{u})$ and $\mathbf{e} = \mathrm{vec}(\mathbf{E})$.

We model the product graph signal $\mathbf{Y} \in \mathbb{R}^{P\times Q}$ as the product graph filtered GP (referred to as PGGP) given by

$$\mathbf{Y} = \mathbf{H}_Q\mathbf{F}\mathbf{H}_P, \tag{7}$$

where $\mathbf{F} \in \mathbb{R}^{P\times Q}$ is the unknown latent variable. We assume a matrix-variate GP prior on $\mathbf{f} = \mathrm{vec}(\mathbf{F})$

$$\mathbf{f} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{Q}_\phi \otimes \mathbf{P}_\rho\right), \tag{8}$$

with $\mathbf{0}$ mean and kernel function $\mathbf{Q}_\phi \otimes \mathbf{P}_\rho$ parameterized by the unknowns ρ and ϕ. Here, the matrices $\mathbf{P}_\rho \in \mathbb{R}^{P\times P}$ and $\mathbf{Q}_\phi \in \mathbb{R}^{Q\times Q}$ are the kernel functions, that measure the similarity between the nodes of product graph factors $\mathcal{G}_P$ and $\mathcal{G}_Q$ using the node features $\mathbf{\Theta}_P \in \mathbb{R}^{D_P\times P}$ and $\mathbf{\Theta}_Q \in \mathbb{R}^{D_Q\times Q}$, respectively. Some popular choices of kernel functions are squared exponential kernel, Matérn kernel, and rational quadratic kernel, to name a few. For the multi-domain

data lies on a regular grid (no graph structure), choosing $\mathbf{H}_P = \mathbf{I}$ and $\mathbf{H}_Q = \mathbf{I}$ boils down to [10]. In contrast, we incorporate the underlying graph structure using generalized product graph filter (3) for multi-domain graph data defined on the product graph.

As the product graph filtering (4) is a linear operation, from (8), the product graph signal $\mathbf{y} = \mathrm{vec}(\mathbf{Y})$ follows a Gaussian distribution as

$$\mathbf{y} \sim \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}) \tag{9}$$

with

$$\begin{aligned}\mathbf{\Sigma} &= \left(\mathbf{H}_Q^T \otimes \mathbf{H}_P\right)\left(\mathbf{Q}_\phi \otimes \mathbf{P}_\rho\right)\left(\mathbf{H}_Q^T \otimes \mathbf{H}_P\right)^T \\ &= \mathbf{\Sigma}_Q \otimes \mathbf{\Sigma}_P,\end{aligned}$$

where $\mathbf{\Sigma}_P = \mathbf{H}_P \mathbf{P}_\rho \mathbf{H}_P^T$ and $\mathbf{\Sigma}_Q = \mathbf{H}_Q^T \mathbf{Q}_\phi \mathbf{H}_Q$. From (6) and (9), the joint PDF of the observed noisy graph signals $\mathbf{x}_\mathrm{o}$ and the true graph signals $\mathbf{y}_\mathrm{u}$ on the observed nodes is

$$\begin{bmatrix} \mathbf{x}_\mathrm{o} \\ \mathbf{y}_\mathrm{u} \end{bmatrix} = \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{\Sigma}_{\mathrm{oo}} & \mathbf{\Sigma}_{\mathrm{ou}} \\ \mathbf{\Sigma}_{\mathrm{ou}}^T & \mathbf{\Sigma}_{\mathrm{uu}} \end{bmatrix}\right), \tag{10}$$

with

$$\begin{aligned}\mathbf{\Sigma}_{\mathrm{oo}} &= \mathbf{S}_Q \mathbf{\Sigma}_Q \mathbf{S}_Q^T \otimes \mathbf{S}_P \mathbf{\Sigma}_P \mathbf{S}_P^T + \sigma^2 \mathbf{I}, \\ \mathbf{\Sigma}_{\mathrm{ou}} &= \mathbf{S}_Q \mathbf{\Sigma}_Q \bar{\mathbf{S}}_Q^T \otimes \mathbf{S}_P \mathbf{\Sigma}_P \bar{\mathbf{S}}_P^T, \\ \mathbf{\Sigma}_{\mathrm{uu}} &= \bar{\mathbf{S}}_Q \mathbf{\Sigma}_Q \bar{\mathbf{S}}_Q^T \otimes \bar{\mathbf{S}}_P \mathbf{\Sigma}_P \bar{\mathbf{S}}_P^T.\end{aligned}$$

Then the predictive posterior PDF of the graph signals on the unobserved nodes of the product graph is $p(\mathbf{y}_\mathrm{u}|\mathbf{x}_\mathrm{o}, \mathbf{L}_P, \mathbf{L}_Q) = \mathcal{N}\left(\hat{\mathbf{y}}_\mathrm{u}, \hat{\mathbf{\Sigma}}_\mathrm{u}\right)$, with mean $\hat{\mathbf{y}}_\mathrm{u}$ and covariance $\hat{\mathbf{\Sigma}}_\mathrm{u}$ given by

$$\begin{aligned}\hat{\mathbf{y}}_\mathrm{u} &= \mathbf{\Sigma}_{\mathrm{ou}}^T \mathbf{\Sigma}_{\mathrm{oo}}^{-1} \mathbf{x}_\mathrm{o}, \\ \hat{\mathbf{\Sigma}}_\mathrm{u} &= \mathbf{\Sigma}_{\mathrm{uu}} - \mathbf{\Sigma}_{\mathrm{ou}}^T \mathbf{\Sigma}_{\mathrm{oo}}^{-1} \mathbf{\Sigma}_{\mathrm{ou}}.\end{aligned} \tag{11}$$

The mean $\hat{\mathbf{y}}_\mathrm{u}$ of the predictive PDF gives us the point estimate of the unobserved graph signals and the diagonal entries of $\hat{\mathbf{\Sigma}}_\mathrm{u}$ that collects the variance captures the model quantify the uncertainty.

The predictive posterior PDF depends on the kernel parameters ρ, ϕ and the filter coefficients $\boldsymbol{\alpha} \in \mathbb{R}^{k_P+1}$, $\boldsymbol{\beta} \in \mathbb{R}^{k_Q+1}$, which are not known and have to be estimated from the observed data. To find the optimal parameters, we maximize the marginal likelihood $p(\mathbf{x}_\mathrm{o}) = \mathcal{N}(\mathbf{0}, \mathbf{\Sigma}_{\mathrm{oo}})$ with respect to the unknown parameters as

$$\underset{\boldsymbol{\alpha},\boldsymbol{\beta},\rho,\phi}{\arg\max} \; -\frac{N}{2}\log(2\pi) - \frac{1}{2}\log|\mathbf{\Sigma}_{\mathrm{oo}}| - \mathbf{x}_\mathrm{o}^T \mathbf{\Sigma}_{\mathrm{oo}}^{-1} \mathbf{x}_\mathrm{o}. \tag{12}$$

The optimization problem in (12) is a non-convex optimization problem. We solve (12) using an accelerated gradient descent method implemented as the ADAM optimizer.

Remark 1. *Ignoring the underlying product graph structure, i.e., replacing the product graph filter with $\mathbf{H}_N \in \mathbb{R}^{N \times N}$ and the structured sampling matrices in (2) with random selection matrices ($\mathbf{S}_N \in \mathbb{R}^{N_\mathrm{o} \times N_\mathrm{o}}$ and $\bar{\mathbf{S}}_N \in \mathbb{R}^{N_\mathrm{u} \times N_\mathrm{u}}$), the PGGP boils down to the graph Gaussian process [8] (referred to as GGP).*

Ignoring the product graph structure, computing the predictive PDF (11) and learning the unknown parameters (12) for the both PGGP and GGP involves inverting a $\mathbf{\Sigma}_{\mathrm{oo}} \in \mathbb{R}^{N_\mathrm{o} \times N_\mathrm{o}}$ matrix, which costs about $\mathcal{O}\left(N_\mathrm{o}^3\right)$ flops. In addition, GGP requires the kernel function in the main memory, which requires $\mathcal{O}(N^2)$ storage. Using the properties of Kronecker product of matrices, PGGP incurs a less computation cost and storage.

From (10), the covariance matrix $\mathbf{\Sigma}_{\mathrm{oo}}$ has a Kronecker structure $\mathbf{\Sigma}_{\mathrm{oo}} = \mathbf{A}_Q \otimes \mathbf{A}_P + \sigma^2 \mathbf{I}$, where $\mathbf{A}_Q = \mathbf{S}_Q \mathbf{\Sigma}_Q \mathbf{S}_Q^T$ and $\mathbf{A}_P \mathbf{S}_P \mathbf{\Sigma}_P \mathbf{S}_P^T$ are symmetric matrices with eigenvalue decomposition $\mathbf{A}_P = \mathbf{U}_P \mathbf{\Lambda}_P \mathbf{U}_P^T$ and $\mathbf{A}_Q = \mathbf{U}_Q \mathbf{\Lambda}_Q \mathbf{U}_Q^T$. Here, $\mathbf{U}_P \in \mathbb{R}^{P_\mathrm{o} \times P_\mathrm{o}}$ and $\mathbf{U}_Q \in \mathbb{R}^{Q_\mathrm{o} \times Q_\mathrm{o}}$ are the eigenvector matrices and $\mathbf{\Lambda}_P \in \mathbb{R}^{P_\mathrm{o} \times P_\mathrm{o}}$ and $\mathbf{\Lambda}_Q \in \mathbb{R}^{Q_\mathrm{o} \times Q_\mathrm{o}}$ are the diagonal eigenvalue matrices. Then, the EVD of $\mathbf{\Sigma}_{\mathrm{oo}} \in \mathbb{R}^{N_\mathrm{o} \times N_\mathrm{o}}$ is $\mathbf{\Sigma}_{\mathrm{oo}} = \mathbf{U}\left(\mathbf{\Lambda} + \sigma^2 \mathbf{I}\right)\mathbf{U}^T$, where $\mathbf{U} = \mathbf{U}_Q \otimes \mathbf{U}_P$ and $\mathbf{\Lambda} = \mathbf{\Lambda}_P \otimes \mathbf{\Lambda}_Q$ are the eigenvector and the diagonal eigenvalue matrices, respectively. We can rewrite $\log|\mathbf{\Sigma}_{\mathrm{oo}}|$ and $\mathbf{\Sigma}_{\mathrm{oo}}^{-1}$ as

$$\begin{aligned}\log|\mathbf{\Sigma}_{\mathrm{oo}}| &= \sum_{i=0}^{N_\mathrm{o}} \log(\lambda_i + \sigma^2) \\ \mathbf{\Sigma}_{\mathrm{oo}}^{-1} &= \mathbf{U}\left(\mathbf{\Lambda} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{U}^T.\end{aligned} \tag{13}$$

Thus the PGGP inference and parameter estimation can be implemented just by computing the EVD of two small matrices, which costs about $\mathcal{O}(P_\mathrm{o}^3 + Q_\mathrm{o}^3)$ flops. Furthermore, storing two kernel functions $\mathbf{P}_\rho \in \mathbb{R}^{P \times P}$ and $\mathbf{Q}_\phi \in \mathbb{R}^{Q \times Q}$ realted to the factor graphs requires $\mathcal{O}(P^2 + Q^2)$ memory.

To summarize, PGGP incurs less computation cost and storage compared to GGP. A comparison of the average run time of PGGP and GGP is given in Fig. 1.

IV. Numerical experiments

In this section, we evaluate the performance of the proposed PGGP approach on synthetic and real-world datasets. We compare the performance of PGGP, which considers the product graph structure, with GGP, which ignores the product graph structure under two different settings. In Section IV-A, we conduct experiments on the data imputation problem. In Section IV-B, we test the PGGP performance on active learning, where the learning algorithm sequentially acquires observations based on a variance obtained from the posterior predictive PDF.

A. Data imputation and classification

In this section, we evaluate the performance of the proposed PGGP method on the data imputation task on synthetic and real-world datasets. Given multi-domain data $\mathbf{Y} \in \mathbb{R}^{P \times Q}$, we view it as a product graph signal indexed by the nodes of a product graph $\mathcal{G}_N$ formed by the Cartesian product of two smaller graphs $\mathcal{G}_P$ and $\mathcal{G}_Q$. We randomly sample the graph data on the subset of nodes, add Gaussian noise of variance $\sigma^2 = 0.5$, and treat it as a noisy graph signal. We use $\mathbf{x}_\mathrm{o}$ to compute the predictive posterior PDF of the graph signals on the remaining nodes (given by mean $\hat{\mathbf{y}}_\mathrm{u} \in \mathbb{R}^{N_\mathrm{u}}$ and covariance matrix $\hat{\mathbf{\Sigma}}_\mathrm{u} \in \mathbb{R}^{N_\mathrm{u} \times N_\mathrm{u}}$) from PGGP and GGP [8]. We vary the percentage of the sampled nodes on each graph

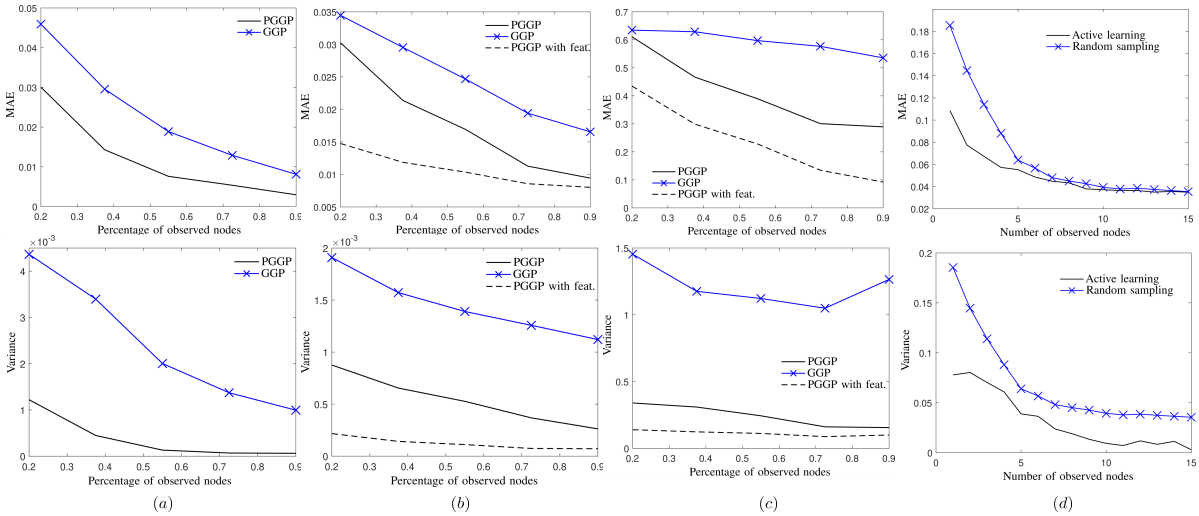

Fig. 2: Performance on synthetic and real-world datasets for GSSL (Figs. $(a)-(c)$) and AL (Fig. (d)) tasks. (a) Synthetic dataset. (b) AQI dataset. (c) Coil dataset. (d) AL on AQI dataset.

factor and report the quality of the estimate in terms of the mean absolute error (MAE) and total variance (variance) of the estimate. Given a true signal $\mathbf{y}_{\mathrm{u}} \in \mathbb{R}^{N_{\mathrm{u}}}$ on the unobserved nodes and its predictive posterior PDF, MAE is defined as $\frac{1}{N_{\mathrm{u}}}\|\mathbf{y}_{\mathrm{u}} - \hat{\mathbf{y}}_{\mathrm{u}}\|_1$, and total variance as $\frac{1}{N_{\mathrm{u}}}\sum_{i=1}^{N_{\mathrm{u}}}\left[\hat{\mathbf{\Sigma}}_{\mathrm{u}}\right]_{i,i}$. For all the experiments, when the features related to the nodes of the graph are available, we choose the squared exponential kernel $k\left(\mathbf{x}_i, \mathbf{x}_j\right) = \exp\left(-0.5\|\mathbf{x}_i - \mathbf{x}_j\|_2^2/\theta^2\right)$, with unknown parameter θ as a kernel function (henceforth, referred to as PPGP with feat.). When the data is not available we choose the kernel as the identity matrix obtained from one hot encoded features (henceforth, referred to as PGGP).

In order to evaluate the performance of PGGP we consider the following synthetic and real-world datasets.

1) Synthetic dataset: We generate a product graph $\mathcal{G}_N$ with $N = 500$ nodes formed by the Cartesian product of the two graphs $\mathcal{G}_P$ and $\mathcal{G}_Q$ with $P = 20$ and $Q = 25$ nodes, respectively. The signal on the product graph $\mathbf{y} \in \mathbb{R}^{500}$ is the obtained by taking random linear combinations of eigenvectors corresponding to the smallest 5 eigenvalues of the normalized graph Laplacian matrix $\mathbf{L}_N \in \mathbb{R}^{500\times500}$. We reshape the product graph signal $\mathbf{y}$ as a matrix $\mathbf{Y} \in \mathbb{R}^{20\times25}$.

2) Coil dataset: Coil data [12] is the collection of images of different objects collected from multiple views (orientations). For our experiments, we consider images of 10 different objects collected from 36 views, where each image is of size 128×128. We view each image as the node of a product graph $\mathcal{G}_N$ formed by the Cartesian product of a object graph $\mathcal{G}_P$ with $P = 10$ nodes and a view graph $\mathcal{G}_Q$ with $Q = 36$ nodes. We view the pixel intensities as the node features, with which form the k nearest neighbors graphs for $\mathcal{G}_P$ and $\mathcal{G}_Q$. We cluster the nodes (images) of the object-view product graph into 6 groups by applying spectral clustering algorithm [13] on the normalized graph Laplacian matrix $\mathbf{L}_N \in \mathbb{R}^{360\times360}$. We treat the cluster membership as the product graph signal $\mathbf{Y} \in \mathbb{R}^{10\times36}$.

3) AQI data: We consider the air quality index (AQI) values and the concentration of different pollutants related to AQI measured across 20 different locations in India for 180 days [14]. We compute the average AQI and concentration of pollutants (particulate matter (pm), carbon monoxide (co) and ozone (o3)) every six days and collect them in the matrices $\mathbf{\Theta}_{\mathrm{AQI}} \in \mathbb{R}^{20\times30}$, and $\mathbf{\Theta}_{\mathrm{pm}} \in \mathbb{R}^{20\times30}$, $\mathbf{\Theta}_{\mathrm{co}} \in \mathbb{R}^{20\times30}$, $\mathbf{\Theta}_{\mathrm{o3}} \in \mathbb{R}^{20\times30}$. We view the space-time measurements of AQI as the product graph signal (i.e., $\mathbf{Y} = \mathbf{\Theta}_{\mathrm{AQI}}$) formed by the Cartesian product of a space graph $\mathcal{G}_P$ and a time graph $\mathcal{G}_Q$ with 20 and 30 nodes, respectively. Furthermore, we assume the concentration of pollutants as node features, of which we use a subset of them to construct the graph factors, and we use the remaining features to learn the kernel parameters. Specifically, we use the columns of the matrices $\mathbf{\Theta}_1 = [\mathbf{\Theta}_{\mathrm{pm}}\, \mathbf{\Theta}_{\mathrm{o3}}]$ and $\mathbf{\Theta}_2 = [\mathbf{\Theta}_{\mathrm{pm}}^T\, \mathbf{\Theta}_{\mathrm{o3}}^T]$ as the features and construct the k nearest neighbor graphs $\mathcal{G}_P$ and $\mathcal{G}_Q$. We use the rows and columns of $\mathbf{\Theta}_{\mathrm{co}}$ (i.e., $\mathbf{\Theta}_P = \mathbf{\Theta}_{\mathrm{co}}$ and $\mathbf{\Theta}_Q = \mathbf{\Theta}_{\mathrm{co}}^T$) as the node features of $\mathcal{G}_P$ and $\mathcal{G}_Q$ to learn the kernel parameters.

In Figs. $1(a), 1(b)$, and $1(c)$, we report the average MAE and variance (averaged over 30 independent realizations of sampling matrices and corresponding noisy observed product graph signals) of the PGGP and GGP methods. The figure shows that the proposed PGGP outperforms the GGP on all the synthetic and real-world datasets. The superior performance of PPGP explains the advantage of taking product graph structure into account in the considered task. Furthermore, as expected, the MAE and total variance of the proposed PGGP decreases as we increase the percentage of observed signals on each graph factor. To demonstrate the importance of node features, we repeat the experiments on PGGP while considering graph structure and node features on AQI and Coil datasets. The results also demonstrate that taking the nodal features significantly improves the performance of the PGGP. To summarize, we emphasize that the proposed PGGP method, which considers the product graph structure, incurs a lower

computational cost than GGP and has a superior peformance on the considered datasets.

B. Active learning

In this section, we evaluate the performance of the proposed PGGP method for active learning (AL) in the PGGP setting. In AL, the learner (or algorithm) starts with one randomly labeled node and sequentially queries the next node to be labeled or observed (from the set of unobserved nodes) based on an acquisition function. By carefully selecting the nodes to be labeled, AL performs better on learning tasks than randomly querying the node labels [7], [15].

In the PGGP setting, we start with an observation from a randomly selected node from each graph factor and compute the predictive posterior PDF of the signals on all the unobserved nodes of the product graph using PGGP. We maximize variance that captures the model uncertainty and select the next product graph node to be observed as

$$r_t = \arg\max_i \ \mathbf{e}_i^T \hat{\mathbf{\Sigma}}_{\mathbf{u}} \mathbf{e}_i, \quad i \in \mathcal{U}_t \tag{14}$$

where $\mathcal{U}_t$ is the set with all the unobserved nodes in the product graph at iteration t and $\mathbf{e}_i \in \mathbb{R}^{N_u}$ is the ith standard basis vector. Here, iteration index t also corresponds to the number of observed nodes in the product graph. As each node in the product graph is indexed by the pair of vertices in each graph factor, the node index corresponds to pth and qth node in its graph factors. We then provide the graph signals related to pth and qth node to the learner and run PGGP inference and learning with all the observed nodes till the current iteration. At each iteration, we evaluate the performance of PGGP in terms of mean and total variance on the remaining unlabelled nodes. In Fig. $2(d)$, we compare the performance of active learning with the random acquisition, where we randomly select the query nodes from graph factors at each iteration on AQI dataset. For this data, we add a noise variance $\sigma^2 = 1$ to the product graph signal $\mathbf{Y}$. The results show that PGGP is data efficient and requires fewer observations.

V. Conclusion

We developed PGGP for multi-domain data imputation and active learning, wherein PGGP incorporates a product graph structure into the Gaussian process kernel. We compute a posterior predictive PDF for imputing missing data whose mean vector and covariance matrices are computed in closed form. We have shown that PGGP incurs less computation cost than GGP. Experimental results on synthetic and real-world datasets on data imputation and active learning corroborate the effectiveness of the proposed method.

References

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

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

[3] S. K. Kadambari and S. P. Chepuri, "Product graph learnings from multi-domain data with sparsity and rank constraints," *IEEE Trans. Signal Process.*, vol. 69, pp. 5665–5680, 2021.

[4] Q. Lu, V. N. Ioannidis, and G. B. Giannakis, "Semi-supervised learning of processes over multi-relational graphs," in *Proc. of the IEEE Int. Conf. on Acoustics, Speech and Signal Process. (ICASSP)*, Barcelona, May 2020.

[5] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *Proc. of the Int. Conf. Lear. Rep.*, Toulon, France, Apr. 2017.

[6] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*. MIT press, 2006.

[7] Y. C. Ng, N. Colombo, and R. Silva, "Bayesian semi-supervised learning with graph Gaussian processes," *Adv. in Nueral Info. Proc. Systems*, vol. 31, 2018.

[8] F. Opolka, Y.-C. Zhi, P. Lio, and X. Dong, "Adaptive Gaussian processes on graphs via spectral graph wavelets," in *Proc. Int. Conf. on Artificial Intelligence and Statistics*, Valencia, Spain, Mar. 2022.

[9] S. Flaxman, A. Wilson, D. Neill, H. Nickisch, and A. Smola, "Fast kronecker inference in Gaussian processes with non-Gaussian likelihoods," in *Proc. of the Int. Conf. Mach. Lear*, vol. 37, Lille, France, July 2015.

[10] Y. Saatçi, "Scalable inference for structured Gaussian process models," Ph.D. dissertation, Citeseer, 2012.

[11] G. Ortiz-Jiménez, M. Coutino, S. P. Chepuri, and G. Leus, "Sparse sampling for inverse problems with tensors," *IEEE Trans. Signal Process.*, vol. 67, no. 12, pp. 3272–3286, June 2019.

[12] S. Nane, S. Nayar, and H. Murase, "Columbia object image library: Coil-20," *Dept. Comp. Sci., Columbia University, New York, Tech. Rep*, Feb. 1996.

[13] U. Von Luxburg, "A tutorial on spectral clustering," *Statistics and computing*, vol. 17, no. 4, pp. 395–416, 2007.

[14] "Central control room for air quality management, India," https://app.cpcbccr.com/.

[15] K. D. Polyzos, Q. Lu, and G. B. Giannakis, "Weighted ensembles for active learning with adaptivity," *arXiv e-prints*, pp. arXiv–2206, 2022.