Online Graph-Adaptive Learning with Scalability and Privacy

Shen, Yanning; Leus, Geert; Giannakis, Georgios B.

DOI
10.1109/TSP.2019.2904922

Publication date
2019

Document Version
Final published version

Published in
IEEE Transactions on Signal Processing

Online Graph-Adaptive Learning With Scalability and Privacy

Yanning Shen, *Student Member, IEEE*, Geert Leus, *Fellow, IEEE*, and Georgios B. Giannakis, *Fellow, IEEE*

***Abstract*—Graphs are widely adopted for modeling complex systems, including financial, biological, and social networks. Nodes in networks usually entail attributes, such as the age or gender of users in a social network. However, real-world networks can have very large size, and nodal attributes can be unavailable to a number of nodes, e.g., due to privacy concerns. Moreover, new nodes can emerge over time, which can necessitate real-time evaluation of their nodal attributes. In this context, this paper deals with scalable learning of nodal attributes by estimating a nodal function based on noisy observations at a subset of nodes. A multikernel-based approach is developed, which is scalable to large-size networks. Unlike most existing methods that re-solve the function estimation problem over all existing nodes whenever a new node joins the network, the novel method is capable of providing real-time evaluation of the function values on newly joining nodes without resorting to a batch solver. Interestingly, the novel scheme only relies on an encrypted version of each node's connectivity in order to learn the nodal attributes, which promotes privacy. Experiments on both synthetic and real datasets corroborate the effectiveness of the proposed methods.**

***Index Terms*—Graph signal reconstruction, kernel-based learning, learning over dynamic graphs, online learning.**

I. Introduction

ESTIMATING nodal functions/signals over networks is a task emerging in various domains, such as social, brain, and power networks, to name a few. Functions of nodes can represent certain attributes or classes of these nodes. In Facebook for instance, each node represents a person, and the presence of an edge indicates that two persons are friends, while nodal attributes can be age, gender or movie ratings of each person. In financial networks, where each node is a company, with links denoting trade between two companies, the function of the node can represent the category that each company belongs to, e.g., technology-, fashion-, or education-related.

In real-world networks, there are often unavailable nodal function values, due to, e.g., privacy issues. Hence, a topic of great practical importance is to interpolate missing nodal values (class, ranking or function), based on the function values at a subset of observed nodes. Interpolation of nodal function values often relies on the assumption of "smoothness" over the graphs, which implies that neighboring nodes will have similar nodal function values. For example, in social networks, people tend to rate e.g., movies similar to their friends, and in financial networks, companies that trade with each other usually belong to the same category. From this point of view, function estimation over graphs based on partial observations has been investigated extensively, [1]–[6]. Function estimation has been also pursued in the context of semi-supervised learning, e.g., for transductive regression or classification, see e.g., [7]–[10]. The same task has been studied recently as signal reconstruction over graphs, see e.g., [11]–[15], where signal values on unobserved nodes can be estimated by properly introducing a graph-aware prior. Kernel-based methods for learning over graphs offer a unifying framework that includes linear and nonlinear function estimators [13], [16], [17]. The nonlinear methods outperform the linear ones but suffer from the curse of dimensionality [18], rendering them less attractive for large-scale networks.

To alleviate this limitation, a scalable kernel-based approach will be introduced in the present paper, which leverages the random feature approximation to ensure *scalability* while also allowing *real-time* evaluation of the functions over large-scale dynamic networks. In addition, the novel approach incorporates a data-driven scheme for *adaptive* kernel selection.

Adaptive learning over graphs has been also investigated for tracking and learning over possibly dynamic networks, e.g., [19], [20]. Least mean-squares and recursive least-squares adaptive schemes have been developed in [19], without explicitly accounting for evolving network topologies. In contrast, [20] proposed a kernel-based reconstruction scheme to track time-varying signals over time-evolving topologies, but assumed that the kernel function is selected a priori. Node2vec [21] and DeepWalk [22] that can sequentially process the graph adjacency matrix have been developed based on deep learning techniques. All these prior works assume that the network size is fixed.

In certain applications however, new nodes may join the network over time. For example, hundreds of new users are joining Facebook or Netflix every day, and new companies are founded in financial networks regularly. Real-time and scalable estimation of the desired functions on these newly-joining nodes is of great importance. While simple schemes such as averaging over one- or multi-hop neighborhoods are scalable to network size by predicting the value on each newly-coming node as a weighted

Manuscript received August 3, 2018; revised December 2, 2018; accepted February 20, 2019. Date of publication March 13, 2019; date of current version April 4, 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gwo Giun Lee. This work was supported in part by the National Science Foundation under NSF 1711471 and 1500713 and in part by the National Institutes of Health under NIH 1R01GM104975-01. *(Corresponding author: Georgios B. Giannakis.)*

Y. Shen and G. B. Giannakis are with the Department of ECE and the Digital Technology Center, University of Minnesota, Minneapolis, MN 55455 USA (e-mail: shenx513@umn.edu; georgios@umn.edu).

G. Leus is with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, 2826 CD Delft, The Netherlands (e-mail: g.j.t.leus@tudelft.nl).

Digital Object Identifier 10.1109/TSP.2019.2904922

combination of its multi-hop neighborhoods [23], they do not capture global information over the network. In addition, existing rigorous approaches are in general less efficient in accounting for newly-joining nodes, and need to solve the problem over all nodes, every time new nodes join the network, which incurs complexity $\mathcal{O}(N^3)$, where N denotes the network size [13], [16]. As a result, these methods are not amenable to real-time evaluation over newly-joining nodes. To this end, the present paper develops a scalable *online graph-adaptive* algorithm that can efficiently estimate nodal functions on newly-joining nodes 'on the fly.' Note that recently GraphSAGE has been developed to cope with possibly growing networks [24].

Besides scalability and adaptivity, nodes may have firm *privacy* requirements, and may therefore not be willing to reveal who their neighbors are. However, most graph-based learning methods require knowing the entire connectivity pattern, and thus cannot meet the privacy requirements, e.g., [21], [22], [24]. The novel random feature based approach on the other hand, only requires an encrypted version of each node's connectivity pattern, which makes it appealing for networks with stringent privacy constraints.

In short, we put forth a novel online multikernel learning (MKL) framework for effectively learning and tracking nonlinear functions over graphs. Our contributions are as follows.

- **c1)** A scalable MKL approach is developed to efficiently estimate the nodal function values both on the observed and un-observed nodes of a graph;
- **c2)** The resultant algorithm is capable of estimating the function value of newly incoming nodes with high accuracy without having to solve the batch problem over all nodes, making it highly scalable as the network size grows, and suitable for nodal function estimation in dynamic networks;
- **c3)** Unlike most existing methods that rely on nodal feature vectors in order to learn the function, the proposed scheme simply capitalizes on the connectivity pattern of each node, while at the same time, nodal feature vectors can be easily incorporated if available; and,
- **c4)** The proposed algorithm does not require nodes to share connectivity patterns. Instead, a privacy-preserving scheme is developed for estimating the nodal function values based on an encrypted version of the nodal connectivity patterns, hence respecting node privacy.

The rest of this paper is organized as follows. Preliminaries are in Section II, while Section III presents an online kernel-based algorithm that allows sequential processing of nodal samples. Section IV develops an online MKL scheme for sequential data-driven kernel selection, which allows graph-adaptive selection of kernel functions to best fit the learning task of interest. Finally, results of corroborating numerical tests on both synthetic and real data are presented in Section VI, while concluding remarks along with a discussion of ongoing and future directions are given in Section VII.

Notation: Bold uppercase (lowercase) letters denote matrices (column vectors), while $(\cdot)^\top$ and $\lambda_i(.)$ stand for matrix transposition, and the ith leading eigenvalue of the matrix argument, respectively. The identity matrix will be represented by $\mathbf{I}$, while $\mathbf{0}$ will denote the matrix of all zeros, and their dimensions will be clear from the context. Finally, the ℓ_p and Frobenius norms will be denoted by $\|\cdot\|_p$, and $\|\cdot\|_F$, respectively.

II. Kernel-Based Learning Over Graphs

Consider a graph $\mathcal{G}(\mathcal{V},\mathcal{E})$ of N nodes, whose topology is captured by a known adjacency matrix $\mathbf{A}\in\mathbb{R}^{N\times N}$. Let $a_{nn'}\in\mathbb{R}$ denote the (n,n') entry of $\mathbf{A}$, which is nonzero only if an edge is present from node n' to n. A real-valued function (or signal) on a graph is a mapping $f:\mathcal{V}\to\mathbb{R}$, where $\mathcal{V}$ is the set of vertices. The value $f(v)=x_v$ represents an attribute of $v\in\mathcal{V}$, e.g., in the context of brain networks, x_{v_n} could represent the sample of an electroencephalogram (EEG), or functional magnetic resonance imaging (fMRI) measurement at region n. In a social network, x_{v_n} could denote the age, political alignment, or annual income of the nth person. Suppose that a collection of noisy samples $\{y_m=x_{v_{n_m}}+e_m\}_{m=1}^M$ is available, where e_m models noise, and $M\leq N$ represents the number of measurements. Given $\{y_m\}_{m=1}^M$, and with the graph topology known, the goal is to estimate $f(v)$, and thus reconstruct the graph signal at unobserved vertices. Letting $\mathbf{y}:=[y_1,\ldots,y_M]^\top$, the observation vector obeys

$$\mathbf{y}=\mathbf{\Psi}\mathbf{x}+\mathbf{e} \tag{1}$$

where $\mathbf{x}:=[x_{v_1},\ldots,x_{v_N}]^\top$, $\mathbf{e}:=[e_1,\ldots,e_M]^\top$, and $\mathbf{\Psi}\in\{0,1\}^{M\times N}$ is a sampling matrix with binary entries $[\mathbf{\Psi}]_{m,n_m}=1$ for $m=1,\ldots,M$, and 0, elsewhere.

Given $\mathbf{\Psi}$, $\mathbf{y}$, and $\mathbf{A}$, the goal is to estimate $\mathbf{x}$ over the entire network. To tackle the under-determined system (1), consider function f belonging to a reproducing kernel Hilbert space (RKHS) defined as [13], [16]

$$\mathcal{H}:=\left\{f:f(v)=\sum_{n=1}^N\alpha_n\kappa(v,v_n),\alpha_n\in\mathbb{R}\right\} \tag{2}$$

where $\kappa:\mathcal{V}\times\mathcal{V}\to\mathbb{R}$ is a pre-selected kernel function. Hereafter, we will let $n_m=m$ for notational convenience, and without loss of generality (wlog). Given $\mathbf{y}$, the RKHS-based estimate is formed as

$$\hat{f}=\arg\min_{f\in\mathcal{H}}\ \frac{1}{M}\sum_{m=1}^M\mathcal{C}(f(v_m),y_m)+\mu\Omega\left(\|f\|_{\mathcal{H}}^2\right) \tag{3}$$

where the cost $\mathcal{C}(\cdot,\cdot)$ can be selected depending on the learning task, e.g., the least-squares (LS) for regression, or the logistic loss for classification; $\|f\|_{\mathcal{H}}^2:=\sum_n\sum_{n'}\alpha_n\alpha_{n'}\kappa(v_n,v_{n'})$ is the RKHS norm; $\Omega(\cdot)$ is an increasing function; and, $\mu>0$ is a regularization parameter that copes with overfitting. According to the definition of graph RKHS in (2), the function estimate can be written as $\hat{f}(v)=\sum_{n=1}^N\alpha_n\kappa(v,v_n):=\bar{\alpha}^\top\bar{\mathbf{k}}(v)$, where $\bar{\boldsymbol{\alpha}}:=[\alpha_1,\ldots,\alpha_N]^\top\in\mathbb{R}^N$ collects the basis coefficients, and $\bar{\mathbf{k}}(v):=[\kappa(v,v_1),\ldots,\kappa(v,v_N)]^\top$. Substituting into the RKHS norm, we find $\|f\|_{\mathcal{H}}^2:=\sum_n\sum_{n'}\alpha_n\alpha_{n'}\kappa(v_n,v_{n'})=\bar{\boldsymbol{\alpha}}^\top\bar{\mathbf{K}}\bar{\boldsymbol{\alpha}}$, where the $N\times N$ kernel matrix $\bar{\mathbf{K}}$ has entries $[\bar{\mathbf{K}}]_{n,n'}:=\kappa(v_n,v_{n'})$; thus, the functional problem (3) boils down to

$$\min_{\bar{\boldsymbol{\alpha}}\in\mathbb{R}^N}\ \frac{1}{M}\sum_{m=1}^M\mathcal{C}(\bar{\boldsymbol{\alpha}}^\top\bar{\mathbf{k}}(v_m),y_m)+\mu\Omega\left(\bar{\boldsymbol{\alpha}}^\top\bar{\mathbf{K}}\bar{\boldsymbol{\alpha}}\right). \tag{4}$$

According to the representer theorem, the optimal solution of (3) admits the finite-dimensional form given by [13], [16]

$$\hat{f}(v) = \sum_{m=1}^{M} \alpha_m \kappa(v, v_m) := \boldsymbol{\alpha}^\top \mathbf{k}(v). \tag{5}$$

where $\boldsymbol{\alpha} := [\alpha_1, \ldots, \alpha_M]^\top \in \mathbb{R}^M$, and $\mathbf{k}(v) := [\kappa(v, v_1), \ldots, \kappa(v, v_M)]^\top$. This means that the coefficients corresponding to the unobserved nodes are all zero. This implies that the function over the graph can be estimated by optimizing over the $M \times 1$ vector $\boldsymbol{\alpha}$ [cf. (3)]

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^M} \ \frac{1}{M} \sum_{m=1}^{M} \mathcal{C}(\boldsymbol{\alpha}^\top \mathbf{k}(v_m), y_m) + \mu \Omega\left(\boldsymbol{\alpha}^\top \mathbf{K} \boldsymbol{\alpha}\right) \tag{6}$$

where $\mathbf{K} := \boldsymbol{\Psi}^\top \bar{\mathbf{K}} \boldsymbol{\Psi}$. For general kernel-based learning tasks, $\bar{\mathbf{K}}$ is formed using the nonlinear functions of pairwise correlations $\kappa(v_n, v_{n'}) = \boldsymbol{\phi}_n^\top \boldsymbol{\phi}_{n'}$, where $\boldsymbol{\phi}_n$ denotes the feature vector of node n, which can collect, for example, the buying history of users on Amazon, or the trading history of companies in financial networks. However, such information may not be available in practice, due to, e.g., privacy concerns. This has motivated the graph-kernel based approaches in [13] and [16], to reconstruct the graph signal when only the network structure is available, and the kernel matrix is selected as a nonlinear function of the graph Laplacian matrix. Specifically, these works mostly consider undirected networks, $\mathbf{A} = \mathbf{A}^\top$.

Given the normalized Laplacian matrix $\mathbf{L} := \mathbf{I} - \mathbf{D}^{-1/2} \mathbf{A} \mathbf{D}^{-1/2}$, with $\mathbf{D} := \text{diag}(\mathbf{A}\mathbf{1})$, and letting $\mathbf{L} := \mathbf{U} \boldsymbol{\Lambda} \mathbf{U}^\top$, the family of graphical kernels is

$$\bar{\mathbf{K}} := r^\dagger(\mathbf{L}) := \mathbf{U} r^\dagger(\boldsymbol{\Lambda}) \mathbf{U}^\top \tag{7}$$

where $r(.)$ is a non-decreasing scalar function of the eigenvalues, and † denotes pseudo-inverse. By selecting $r(.)$, different graph properties can be accounted for, including smoothness, band-limitedness, the random walk [16], and diffusion [2].

Although graph-kernel based methods are effective in reconstructing signals over graphs, it can be observed from (7) that formulating $\bar{\mathbf{K}}$ generally requires an eigenvalue decomposition of $\mathbf{L}$, which incurs complexity $\mathcal{O}(N^3)$ that can be prohibitive for large-scale networks. Moreover, even though nodal feature vectors $\{\phi_n\}$ are not necessary to form $\bar{\mathbf{K}}$, the graph-kernel-based scheme requires knowledge of the topology, meaning $\mathbf{A}$, in order to estimate the nodal function of each node. However, in networks with strict privacy requirements, nodes may not be willing to share such information with others. In Facebook, for example, most people do not make their friend list public. In addition, solving (4) assumes that all sampled nodes are available in batch, which may not be true in scenarios where nodes are sampled in a sequential fashion.

In response to these challenges, an online scalable kernel-based method will be developed in the ensuing section to deal with sequentially obtained data samples, over generally dynamic networks. The resultant algorithm only requires encrypted versions of the nodal connectivity patterns of other nodes, and hence it offers privacy.

III. Online Kernel-Based Learning Over Graphs

Instead of resorting to a graph kernel that requires an eigenvalue decomposition of $\mathbf{L}$ in (7), the present section advocates treating the *connectivity pattern of each node as its feature vector*, which can be the nth column $\mathbf{a}_n^{(c)}$ and possibly the nth row $(\mathbf{a}_n^{(r)})^\top$ of the adjacency (if $\mathbf{A}$ is nonsymmetric). We will henceforth term this the *connectivity pattern* of v_n, and denote it as $\mathbf{a}_n$, for brevity. Given $\mathbf{a}_n$, we will interpolate unavailable nodal function values $\hat{f}(v_n)$ using a nonparametric approach, that is different and scalable relative to [16] and [13]. The kernel matrix is now

$$[\bar{\mathbf{K}}]_{n,n'} = \kappa(v_n, v_{n'}) = \kappa(\mathbf{a}_n, \mathbf{a}_{n'}). \tag{8}$$

Again, with M nodes sampled, the representer theorem asserts that the sought function estimator has the form [18]

$$\hat{f}(v_n) = \hat{f}(\mathbf{a}_n) = \sum_{m=1}^{M} \alpha_m \kappa(\mathbf{a}_m, \mathbf{a}_n) := \boldsymbol{\alpha}^\top \mathbf{k}(\mathbf{a}_n) \tag{9}$$

where $\mathbf{k}(\mathbf{a}_n) := [\kappa(\mathbf{a}_n, \mathbf{a}_1) \ldots \kappa(\mathbf{a}_n, \mathbf{a}_M)]^\top$. It can be observed from (9) that $\hat{f}(v_n)$ involves the adjacency of the entire network, namely $\{\mathbf{a}_m\}_{m=1}^M$, which leads to potentially growing complexity $\mathcal{O}(M^3)$ as the number of sampled nodes increases [18].

A. Batch RF-Based Learning Over Graphs

To bypass this growing complexity, we will resort to the so-called random feature approximation [25] in order to reduce the original functional learning task in (4) to a problem with the number of unknown parameters not growing with M. We first approximate κ in (5) using random features (RFs) [25], [26] that are obtained from a shift-invariant kernel satisfying $\kappa(\mathbf{a}_n, \mathbf{a}_{n'}) = \kappa(\mathbf{a}_n - \mathbf{a}_{n'})$. For $\kappa(\mathbf{a}_n - \mathbf{a}_{n'})$ absolutely integrable, its Fourier transform $\pi_\kappa(\mathbf{v})$ exists and represents the power spectral density, which upon normalizing to ensure $\kappa(\mathbf{0}) = 1$, can also be viewed as a probability density function (pdf); hence,

$$\begin{aligned} \kappa(\mathbf{a}_n - \mathbf{a}_{n'}) &= \int \pi_\kappa(\mathbf{v}) e^{j\mathbf{v}^\top(\mathbf{a}_n - \mathbf{a}_{n'})} d\mathbf{v} \\ &:= \mathbb{E}_{\mathbf{v}}\left[e^{j\mathbf{v}^\top(\mathbf{a}_n - \mathbf{a}_{n'})}\right] \end{aligned} \tag{10}$$

where the last equality is due to the definition of the expected value. Drawing a sufficient number of D independent and identically distributed samples $\{\mathbf{v}_i\}_{i=1}^D$ from $\pi_\kappa(\mathbf{v})$, the ensemble mean (10) can be approximated by the sample average

$$\hat{\kappa}(\mathbf{a}_n, \mathbf{a}_{n'}) = \mathbf{z}_{\mathbf{V}}^\top(\mathbf{a}_n) \mathbf{z}_{\mathbf{V}}(\mathbf{a}_{n'}) \tag{11}$$

where $\mathbf{V} := [\mathbf{v}_1, \ldots, \mathbf{v}_D]^\top \in \mathbb{R}^{D \times N}$, and $\mathbf{z}_{\mathbf{V}}$ denotes the $2D \times 1$ *real-valued* RF vector

$$\begin{aligned} \mathbf{z}_{\mathbf{V}}(\mathbf{a}) &= D^{-\frac{1}{2}} \\ &\times \left[\sin(\mathbf{v}_1^\top \mathbf{a}), \ldots, \sin(\mathbf{v}_D^\top \mathbf{a}), \cos(\mathbf{v}_1^\top \mathbf{a}), \ldots, \cos(\mathbf{v}_D^\top \mathbf{a})\right]^\top. \end{aligned} \tag{12}$$

Taking expectations in (11) and using (10), one can verify that $\mathbb{E}_{\mathbf{v}}[\hat{\kappa}(\mathbf{a}_n, \mathbf{a}_{n'})] = \kappa(\mathbf{a}_n, \mathbf{a}_{n'})$, which means $\hat{\kappa}$ is unbiased. Note

that finding $\pi_\kappa(\mathbf{v})$ requires an N-dimensional Fourier transform of κ, which in general requires numerical integration. Nevertheless, it has been shown that for a number of popular kernels, $\pi_\kappa(\mathbf{v})$ is available in closed form [25]. Taking the Gaussian kernel as an example, where $\kappa(\mathbf{a}_n, \mathbf{a}_{n'}) = \exp\left(\|\mathbf{a}_n - \mathbf{a}_{n'}\|_2^2/(2\sigma^2)\right)$, it has a Fourier transform corresponding to the pdf $\mathcal{N}(0, \sigma^{-2}\mathbf{I})$.

Hence, the function that is optimal in the sense of (3) can be cast to a function approximant over the $2D$-dimensional RF space (cf. (9) and (11))

$$\hat{f}^{\mathrm{RF}}(\mathbf{a}) = \sum_{m=1}^{M} \alpha_m \mathbf{z}_{\mathbf{V}}^{\top}(\mathbf{a}_m)\mathbf{z}_{\mathbf{V}}(\mathbf{a}) := \boldsymbol{\theta}^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a}) \tag{13}$$

where $\boldsymbol{\theta}^{\top} := \sum_{m=1}^{M} \alpha_m \mathbf{z}_{\mathbf{V}}^{\top}(\mathbf{a}_m)$. While $\hat{f}$ in (5) is the superposition of nonlinear functions κ, its RF approximant $\hat{f}^{\mathrm{RF}}$ in (13) is a linear function of $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_i)$. As a result, (3) reduces to

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{2D}} \frac{1}{M}\sum_{m=1}^{M} \mathcal{C}(\boldsymbol{\theta}^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_m), y_m) + \mu\Omega\left(\|\boldsymbol{\theta}\|^2\right) \tag{14}$$

where $\|\boldsymbol{\theta}\|^2 := \sum_t \sum_\tau \alpha_t\alpha_\tau \mathbf{z}_{\mathbf{V}}^{\top}(\mathbf{a}_t)\mathbf{z}_{\mathbf{V}}(\mathbf{a}_\tau) \simeq \|f\|_{\mathcal{H}}^2$. A batch solver of (14) has complexity $\mathcal{O}(MD^3)$ that does not grow with N. This batch RF-based approach scales linearly with the number of measured nodes M, and the number of variables is $2D$, which does not depend on M. This allows us to pursue an online implementation as elaborated next.

B. Online RF-Based Learning Over Graphs

Here, we will further leverage RF-based learning over graphs to enable real-time learning and reconstruction of signals evolving over possibly dynamic networks. A scalable online algorithm will be introduced, which can adaptively handle sequentially sampled nodal features and update the sought function estimates.

Training sequentially. In the training phase, we are given a network of N nodes, and the nodal function is sampled in a sequential fashion. Letting v_t denote the node sampled at the tth time slot, and having available $\{\mathbf{a}_t, y_t\}$ at v_t, the online inference task can be written as [cf. (14)]

$$\min_{\boldsymbol{\theta}\in\mathbb{R}^{2D}} \sum_{\tau=1}^{t} \mathcal{L}\left(\boldsymbol{\theta}^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_\tau), y_\tau\right)$$

$$\mathcal{L}\big(\boldsymbol{\theta}^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t), y_t\big) := \mathcal{C}\big(\boldsymbol{\theta}^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t), y_t\big) + \mu\Omega\big(\|\boldsymbol{\theta}\|^2\big). \tag{15}$$

We will solve (15) using online gradient descent [27]. Obtaining v_t per slot t, the RF of its connectivity pattern $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t)$ is formed as in (12), and $\boldsymbol{\theta}_{t+1}$ is updated 'on the fly,' as

$$\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t - \eta_t \nabla\mathcal{L}(\boldsymbol{\theta}_t^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t), y_t) \tag{16}$$

where $\{\eta_t\}$ is the sequence of stepsizes that can tune learning rates. In this paper, we will adopt $\eta_t = \eta$ for simplicity. Iteration (16) provides *a functional update* since $\hat{f}_t^{\mathrm{RF}}(\mathbf{a}) = \boldsymbol{\theta}_t^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a})$. The per-iteration complexity of (16) is $\mathcal{O}(D)$, and $\mathcal{O}(MD)$ for the entire training process, which scales better than $\mathcal{O}(MD^3)$ that is required for a batch solver of (14).

Algorithm 1: Online Kernel Based Learning Over Graphs.

Input: step size $\eta > 0$, and number of RFs D.
Initialization: $\boldsymbol{\theta}_1 = \mathbf{0}$.
Training:
for $t = 1, 2, \ldots, M$ **do**
  Obtain the adjacency vector $\mathbf{a}_t$ of sampled node v_t.
  Construct $\mathbf{z}_p(\mathbf{a}_t)$ via (12) using κ.
  Update $\boldsymbol{\theta}_{t+1}$ via (16).
end for
Inference:
  Construct random feature vector $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_j)$ via (12)
  Infer $\hat{f}(v_j) = \boldsymbol{\theta}_{M+1}^{\top}\mathbf{z}_{\mathbf{V}}(v_j),\ j \in \Omega$.
Accounting for newly-coming node
  Construct random feature vector $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_{\mathrm{new}})$ via (12)
  Estimate $\hat{f}(v_{\mathrm{new}}) = \boldsymbol{\theta}_{M+1}^{\top}\mathbf{z}_{\mathbf{V}}(v_{\mathrm{new}})$.
  If y_{new} available, Update $\boldsymbol{\theta}$ via (16).

Inferring unavailable nodal values. After the training phase, the nodal function value over the un-sampled nodes can be readily estimated by [cf. (13)]

$$\hat{f}(v_i) = \hat{\boldsymbol{\theta}}^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_i),\ \ \forall i \in \mathcal{S}^c \tag{17}$$

where $\hat{\boldsymbol{\theta}}$ is the final estimate after the training phase, i.e., $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}_{M+1}$, and $\mathcal{S}^c$ denotes the index set of the nodes whose signal values have not been sampled in the training phase.

Newly-joining nodes. When new nodes join the network, batch graph-kernel based approaches must expand $\bar{\mathbf{K}}$ in (7) by one row and one column, and re-solve (6) in order to form signal estimates for the newly-joining nodes. Hence, each newly joining node will incur complexity $\mathcal{O}(N^3)$. The novel online RF method on the other hand, can simply estimate the signal on the newly coming node via $\hat{f}(v_{\mathrm{new}}) = \hat{\boldsymbol{\theta}}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_{\mathrm{new}})$, where $\mathbf{a}_{\mathrm{new}} \in \mathbb{R}^N$ denotes the connectivity pattern of the new node with the *existing* nodes in the network. This leads to a complexity of $\mathcal{O}(ND)$ per new node. If in addition, y_{new} is available, then the function estimate can also be efficiently updated via (16) and (13) using $\mathbf{a}_{\mathrm{new}}$ and y_{new}.

The steps of our online RF-based method are summarized in Algorithm 1. A couple of standard learning tasks where Algorithm 1 comes handy are now in order.

Nonlinear regression over graphs: Consider first nonlinear regression over graphs, where the goal is to find a nonlinear function $f \in \mathcal{H}$, such that $y_n = f(v_n) + e_n = f(\mathbf{a}_n) + e_n$ given the graph adjacency matrix $\mathbf{A}$. The criterion is to minimize the regularized prediction error of y_n, typically using the online LS loss $\mathcal{L}(f(\mathbf{a}_t), y_t) := [y_t - f(\mathbf{a}_t)]^2 + \mu\|f\|_{\mathcal{H}}^2$ in (15), whose gradient is (cf. (16))

$$\nabla\mathcal{L}\left(\boldsymbol{\theta}_t^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t), y_t\right) = 2[\boldsymbol{\theta}_t^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t) - y_t]\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t) + 2\mu\boldsymbol{\theta}_t.$$

In practice, y_t can represent a noisy version of each node's real-valued attribute, e.g., temperature in a certain city, and the graph can be constructed based on Euclidean distances among cities. For a fair comparison with alternatives, only the regression task will be tested in the numerical section of this paper.

Nonlinear classification over graphs: We can also handle kernel-based perceptron and kernel-based logistic regression, which aim at learning a nonlinear classifier that best approximates either y_n, or, the pdf of y_n conditioned on $\mathbf{a}_n$. With binary labels $\{\pm 1\}$, the perceptron solves (3) with $\mathcal{L}(f(\mathbf{a}_t), y_t) = \max(0, 1 - y_t f(\mathbf{a}_t)) + \mu \|f\|^2_{\mathcal{H}}$, which equals zero if $y_t = f(\mathbf{a}_t)$, and otherwise equals 1. In this case, the gradient of the presented online RF-based method is (cf. (16))

$$\nabla \mathcal{L}\left(\boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_t), y_t\right) = -2y_t \mathcal{C}(\boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_t), y_t)\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t) + 2\mu\boldsymbol{\theta}_t.$$

Accordingly, given $\mathbf{x}_t$, logistic regression postulates that $\Pr(y_t = 1|\mathbf{x}_t) = 1/(1 + \exp(f(\mathbf{x}_t)))$. Here the gradient takes the form (cf. (16))

$$\nabla \mathcal{L}\left(\boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_t), y_t\right) = \frac{2y_t \exp(y_t \boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_t))}{1 + \exp(y_t \boldsymbol{\theta}_t^\top \mathbf{z}_{\mathbf{V}}(\mathbf{a}_t))} \mathbf{z}_p(\mathbf{x}_t) + 2\mu\boldsymbol{\theta}_t.$$

Classification over graphs arises in various scenarios, where y_n may represent categorical attributes such as gender, occupation or, nationality of users in a social network.

Remark 1 (Privacy): Note that the update in (16) does not require access to $\mathbf{a}_t$ directly. Instead, the only information each node needs to reveal is $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t)$ for each $\mathbf{a}_t$, which involves $\{\sin(\mathbf{a}_t^\top \mathbf{v}_j), \cos(\mathbf{a}_t^\top \mathbf{v}_j)\}_{j=1}^D$. Being noninvertible, these co-sinusoid functions involved in generating the $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t)$ can be viewed as an encryption of the nodal connectivity pattern, which means that given $\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t)$, vector $\mathbf{a}_t$ cannot be uniquely deciphered. Hence, Algorithm 1 preserves privacy.

Remark 2 (Directed graphs): It can be observed from (7) that for $\bar{\mathbf{K}}$ to be a valid kernel, graph-kernel based methods require $\mathbf{A}$, and henceforth $\mathbf{L}$ to be symmetric, which implies they can only directly deal with symmetric/undirected graphs. Such a requirement is not necessary for our RF-based method.

Remark 3 (Time-varying graphs): Real-world networks may vary over time, as edges may disappear or appear. To cope with such changing topologies, the original graph-kernel method needs to recalculate the kernel matrix, and resolve the batch problem whenever one edge changes. In contrast, our online RF-based method can simply re-estimate the nodal values on the two ends of the (dis)appeared edge using (13) with their current $\{\mathbf{a}_n\}$.

Evidently, the performance of Algorithm 1 depends on κ that is so far considered known. As the "best" performing κ is generally unknown and application dependent, it is prudent to adaptively select kernels by superimposing multiple kernels from a prescribed dictionary, as we elaborate next.

IV. Online Graph-Adaptive MKL

In the present section, we develop an online **gr**aph-**ad**aptive learning approach that relies on **ra**ndom features, and leverages multi-**ker**nel approximation to estimate the desired f based on sequentially obtained nodal samples over the graph. The proposed method is henceforth abbreviated as **Gradraker**.

The choice of κ is critical for the performance of single kernel based learning over graphs, since different kernels capture different properties of the graph, and thus lead to function estimates of variable accuracy [13]. To deal with this, combinations of kernels from a preselected dictionary $\{\kappa_p\}_{p=1}^P$ can be employed in (3); see also [13], [26]. Each combination belongs to the convex hull $\bar{\mathcal{K}} := \{\bar{\kappa} = \sum_{p=1}^P \bar{\alpha}_p \kappa_p,\ \bar{\alpha}_p \geq 0,\ \sum_{p=1}^P \bar{\alpha}_p = 1\}$. With $\bar{\mathcal{H}}$ denoting the RKHS induced by $\bar{\kappa} \in \bar{\mathcal{K}}$, one then solves (3) with $\mathcal{H}$ replaced by $\bar{\mathcal{H}} := \mathcal{H}_1 \bigoplus \cdots \bigoplus \mathcal{H}_P$, where $\{\mathcal{H}_p\}_{p=1}^P$ represent the RKHSs corresponding to $\{\kappa_p\}_{p=1}^P$ [28].

The candidate function $\bar{f} \in \bar{\mathcal{H}}$ is expressible in a separable form as $\bar{f}(\mathbf{a}) := \sum_{p=1}^P \bar{f}_p(\mathbf{a})$, where $\bar{f}_p(\mathbf{a})$ belongs to $\mathcal{H}_p$, for $p \in \mathcal{P} := \{1, \ldots, P\}$. To add flexibility per kernel in our ensuing online MKL scheme, we let wlog $\{\bar{f}_p = w_p f_p\}_{p=1}^P$, and seek functions of the form

$$f(v) = f(\mathbf{a}) := \sum_{p=1}^P \bar{w}_p f_p(\mathbf{a}) \in \bar{\mathcal{H}} \tag{18}$$

where $f := \bar{f}/\sum_{p=1}^P w_p$, and the normalized weights $\{\bar{w}_p := w_p/\sum_{p=1}^P w_p\}_{p=1}^P$ satisfy $\bar{w}_p \geq 0$, and $\sum_{p=1}^P \bar{w}_p = 1$. Exploiting separability jointly with the RF-based function approximation per kernel, the MKL task can be reformulated, after letting $\mathcal{L}_t(\hat{f}_p^{\mathrm{RF}}(\mathbf{a}_t)) := \mathcal{L}\big(\boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}_p}(\mathbf{a}_t), y_t\big)$ in (15), as

$$\min_{\{\bar{w}_p\}, \{\hat{f}_p^{\mathrm{RF}}\}} \quad \sum_{t=1}^T \sum_{p=1}^P \bar{w}_p \mathcal{L}_t\left(\hat{f}_p^{\mathrm{RF}}(\mathbf{a}_t)\right) \tag{19a}$$

$$\text{s. to} \quad \sum_{p=1}^P \bar{w}_p = 1,\ \bar{w}_p \geq 0,\ p \in \mathcal{P}, \tag{19b}$$

$$\hat{f}_p^{\mathrm{RF}} \in \left\{\hat{f}_p(\mathbf{a}_t) = \boldsymbol{\theta}^\top \mathbf{z}_{\mathbf{V}_p}(\mathbf{a}_t)\right\},\ p \in \mathcal{P} \tag{19c}$$

which can be solved efficiently 'on-the-fly.' Relative to (14), we replaced M by T to introduce the notion of time, and stress the fact that the nodes are sampled sequentially.

Given the connectivity pattern $\mathbf{a}_t$ of the tth sampled node v_t, an RF vector $\mathbf{z}_p(\mathbf{a}_t)$ is generated per p from the pdf $\pi_{\kappa_p}(\mathbf{v})$ via (12), where $\mathbf{z}_p(\mathbf{a}_t) := \mathbf{z}_{\mathbf{V}_p}(\mathbf{a}_t)$ for notational brevity. Hence, per kernel κ_p and node sample t, we have [cf. (13)]

$$\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{a}_t) = \boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{a}_t) \tag{20}$$

and as in (16), $\boldsymbol{\theta}_{p,t}$ is updated via

$$\boldsymbol{\theta}_{p,t+1} = \boldsymbol{\theta}_{p,t} - \eta \nabla \mathcal{L}(\boldsymbol{\theta}_{p,t}^\top \mathbf{z}_p(\mathbf{a}_t), y_t) \tag{21}$$

with $\eta \in (0, 1)$ chosen constant to effect the adaptation. As far as $\bar{w}_{p,t}$ is concerned, since it resides on the probability simplex, a multiplicative update is well motivated as discussed also in, e.g., [26], [27], [29]. For the un-normalized weights, this update is available in closed form as [26]

$$w_{p,t+1} = w_{p,t} \exp\left(-\eta \mathcal{L}_t\left(\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{a}_t)\right)\right). \tag{22}$$

Having found $\{w_{p,t}\}$ as in (22), the normalized weights in (18) are obtained as $\bar{w}_{p,t} := w_{p,t}/\sum_{p=1}^P w_{p,t}$. Note from (22) that when $\hat{f}_{p,t}^{\mathrm{RF}}$ has a larger loss relative to other $\hat{f}_{p',t}^{\mathrm{RF}}$ with $p' \neq p$ for the tth sampled node, the corresponding $w_{p,t+1}$ decreases more than the other weights. In other words, a more accurate approximant tends to play a more important role in predicting

Algorithm 2: Gradraker Algorithm.

Input: Kernels $\kappa_p,\ p=1,\ldots,P$, step size $\eta>0$, and number of RFs D.
Initialization: $\boldsymbol{\theta}_{p,1}=\mathbf{0}$.
Training:
for $t=1,2,\ldots,T$ **do**
  Obtain the adjacency vector $\mathbf{a}_t$ of node v_t.
  Construct $\mathbf{z}_p(\mathbf{a}_t)$ via (12) using κ_p for $p=1,\ldots,P$.
  Predict $\hat{f}_t^{\mathrm{RF}}(\mathbf{a}_t)=\sum_{p=1}^{P}\bar{w}_{p,t}\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{a}_t)$
  Observe loss function $\mathcal{L}_t$, incur $\mathcal{L}_t(\hat{f}_t^{\mathrm{RF}}(\mathbf{a}_t))$.
  for $p=1,\ldots,P$ **do**
    Obtain loss $\mathcal{L}(\boldsymbol{\theta}_{p,t}^{\top}\mathbf{z}_p(\mathbf{a}_t),y_t)$ or $\mathcal{L}_t(\hat{f}_{p,t}^{\mathrm{RF}}(\mathbf{a}_t))$.
    Update $\boldsymbol{\theta}_{p,t+1}$ and $w_{p,t+1}$ via (21) and (22).
  end for
end for
Inference:
  Construct RF vector $\{\mathbf{z}_p(\mathbf{a}_j)\}$ using $\{\kappa_p\}$.
  Infer $\hat{f}(v_j)=\sum_{p=1}^{P}\bar{w}_{p,T+1}\boldsymbol{\theta}_{p,T+1}^{\top}\mathbf{z}_p(v_j)$.
Accounting for newly-coming node
  Construct RF vector $\{\mathbf{z}_p(\mathbf{a}_{\mathrm{new}})\}$ using $\{\kappa_p\}$.
  Estimate $\hat{f}(v_{\mathrm{new}})=\sum_{p=1}^{P}\bar{w}_{p,T+1}\boldsymbol{\theta}_{p,T+1}^{\top}\mathbf{z}_p(v_{\mathrm{new}})$.
  If y_{new} available update $\{\boldsymbol{\theta}_p,w_p\}$ via (21) and (22).

the ensuing sampled node. In summary, our Gradraker for online graph MKL is listed as Algorithm 2.

Remark 4 (Comparison with batch MKL): A batch MKL based approach for signal reconstruction over graphs was developed in [13]. It entails an iterative algorithm whose complexity grows with N in order to jointly estimate the nodal function, and to adaptively select the kernel function. When new nodes join the network, [13] re-calculates the graphical kernels and re-solves the overall batch problem, which does not scale with the network size. In addition, [13] is not privacy preserving in the sense that in order to estimate the function at any node, one needs to have access to the connectivity pattern of the entire network.

Remark 5 (Comparison with k-NN): An intuitive yet efficient way to predict function values of a newly joining node is to simply combine the values of its k nearest neighbors (k-NN) [23], [30]. Efficient as it is, k-NN faces several challenges: a) At least one of the neighbors must be labeled, which does not always hold in practice, and is not required by the Gradraker; and b) k-NN can only account for local information, while the Gradraker takes also into account the global information of the graph.

A. Generalizations

So far, it is assumed that each node n only has available its own connectivity feature vector $\mathbf{a}_n$. This allows Gradraker to be applied even when limited information is available about the nodes, which many existing algorithms that rely on nodal features cannot directly cope with.

If additional feature vectors $\{\boldsymbol{\phi}_{i,n}\}_{i=1}^{I}$ are actually available per node n other than its own $\mathbf{a}_n$, it is often not known a priori which set of features is the most informative for estimating the signal of interest on the graph. To this end, the novel Gradraker can be adapted by treating the functions learned from different sets of features as *an ensemble of learners*, and combine them in a similar fashion as in (18), that is,

$$f(v_n)=\sum_{i=1}^{I}\beta_i f_i(\boldsymbol{\phi}_{i,n}) \tag{23}$$

Applications to several practical scenarios are discussed in the following.

Semi-private networks. In practice, a node may tolerate sharing its links to its neighbors, e.g., users of Facebook may share their friends-list with friends. In this scenario, each node not only knows its own neighbors, but also has access to who are its neighbors' neighbors, i.e., two-hop neighbors. Specifically, node n has access to $\mathbf{a}_n$, as well as to the nth column of $\mathbf{A}^{(2)}:=\mathbf{AA}$ [1], and a learner $f_2(\boldsymbol{\phi}_{2,n})$ can henceforth be introduced and combined in (23). Moreover, when nodes are less strict about privacy, e.g., when a node is willing to share its multi-hop neighbors, more learners can be introduced and combined 'on the fly' by selecting $\boldsymbol{\phi}_{i,n}$ as the nth column of $\mathbf{A}^{(i)}$ in (23).

Multilayer networks. Despite their popularity, ordinary networks are often inadequate to describe increasingly complex systems. For instance, modeling interactions between two individuals using a single edge can be a gross simplification of reality. Generalizing their *single-layer* counterparts, *multilayer networks* allow nodes to belong to N_g groups, called layers [31], [32]. These layers could represent different attributes or characteristics of a complex system, such as temporal snapshots of the same network, or different types of groups in social networks (family, soccer club, or work related). Furthermore, multilayer networks are able to model systems that typically cannot be represented by traditional graphs, such as heterogeneous information networks [33], [34]. To this end, Gradraker can readily incorporate the information collected from heterogenous sources, e.g., connectivity patterns $\{\mathbf{A}_i\}_{i=1}^{N_g}$ from different layers, by adopting a kernel based learner $f_i(\mathbf{a}_{i,n})$ on the ith layer and combining them as in (23).

Nodal features available. In certain cases, nodes may have nodal features [1] in addition to their $\{\mathbf{a}_n\}$. For example, in social networks, other than the users' connectivity patterns, we may also have access to their shopping history on Amazon. In financial networks, in addition to the knowledge of trade relationships with other companies, there may be additional information available per company, e.g., the number of employees, category of products the company sales, or the annual profit. Gradraker can also incorporate this information by introducing additional learners based on the nodal feature vectors, and combine them as in (23).

V. Performance Analysis

To analyze the performance of the novel Gradraker algorithm, we assume that the following are satisfied.

(as1) *For all sampled nodes $\{v_t\}_{t=1}^{T}$, the loss function $\mathcal{L}(\boldsymbol{\theta}^{\top}\mathbf{z}_{\mathbf{V}}(\mathbf{a}_t),y_t)$ in (15) is convex w.r.t. $\boldsymbol{\theta}$.*

(as2) *For $\boldsymbol{\theta}$ belonging to a bounded set $\boldsymbol{\Theta}$ with $\|\boldsymbol{\theta}\| \leq C_\theta$, the loss is bounded; that is, $\mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_\mathbf{V}(\mathbf{a}_t), y_t) \in [-1, 1]$, and has bounded gradient, meaning, $\|\nabla \mathcal{L}(\boldsymbol{\theta}^\top \mathbf{z}_\mathbf{V}(\mathbf{a}_t), y_t)\| \leq L$.*

(as3) *The kernels $\{\kappa_p\}_{p=1}^P$ are shift-invariant, standardized, and bounded, that is, $\kappa_p(\mathbf{a}_n, \mathbf{a}_{n'}) \leq 1$, $\forall \mathbf{a}_n, \mathbf{a}_{n'}$; and w.l.o.g. they also have bounded entries, meaning $\|\mathbf{a}_n\| \leq 1, \forall n$.*

Convexity of the loss under (as1) is satisfied by the popular loss functions including the square loss and the logistic loss. As far as (as2), it ensures that the losses, and their gradients are bounded, meaning they are L-Lipschitz continuous. While boundedness of the losses commonly holds since $\|\boldsymbol{\theta}\|$ is bounded, Lipschitz continuity is also not restrictive. Considering kernel-based regression as an example, the gradient is $(\boldsymbol{\theta}^\top \mathbf{z}_\mathbf{V}(\mathbf{x}_t) - y_t)\mathbf{z}_\mathbf{V}(\mathbf{x}_t) + \lambda\boldsymbol{\theta}$. Since the loss is bounded, e.g., $\|\boldsymbol{\theta}^\top \mathbf{z}_\mathbf{V}(\mathbf{x}_t) - y_t\| \leq 1$, and the RF vector in (12) can be bounded as $\|\mathbf{z}_\mathbf{V}(\mathbf{x}_t)\| \leq 1$, the constant is $L := 1 + \lambda C_\theta$ using the Cauchy-Schwartz inequality. Kernels satisfying the conditions in (as3) include Gaussian, Laplacian, and Cauchy [25]. In general, (as1)–(as3) are standard in online convex optimization (OCO) [27], [35], and in kernel-based learning [25], [36], [37].

In order to quantify the performance of Gradraker, we resort to the static regret metric, which quantifies the difference between the aggregate loss of an OCO algorithm, and that of the best fixed function approximant in hindsight, see also e.g., [27], [35]. Specifically, for a sequence $\{\hat{f}_t\}$ obtained by an online algorithm $\mathcal{A}$, its static regret is

$$\mathrm{Reg}^{\mathrm{s}}_{\mathcal{A}}(T) := \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_t(\mathbf{a}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(f^*(\mathbf{a}_t)) \tag{24}$$

where $\hat{f}_t^{\mathrm{RF}}$ will henceforth be replaced by $\hat{f}_t$ for notational brevity; and, $f^*(\cdot)$ is defined as the batch solution

$$f^*(\cdot) \in \arg \min_{\{f_p^*,\, p\in\mathcal{P}\}} \sum_{t=1}^{T} \mathcal{L}_t(f_p^*(\mathbf{a}_t))$$
$$\text{with} \quad f_p^*(\cdot) \in \arg\min_{f\in\mathcal{F}_p} \sum_{t=1}^{T} \mathcal{L}_t(f(\mathbf{a}_t)) \tag{25}$$

where $\mathcal{F}_p := \mathcal{H}_p$, with $\mathcal{H}_p$ representing the RKHS induced by κ_p. We establish the regret of our Gradraker approach in the following lemma.

Lemma 1: Under (as1), (as2), and with $\hat{f}_p^*$ defined as $\hat{f}_p^*(\cdot) \in \arg\min_{f\in\hat{\mathcal{F}}_p} \sum_{t=1}^{T} \mathcal{L}_t(f(\mathbf{a}_t))$, with $\hat{\mathcal{F}}_p := \{\hat{f}_p | \hat{f}_p(\mathbf{a}) = \boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{a}),\ \forall \boldsymbol{\theta} \in \mathbb{R}^{2D}\}$, for any p, the sequences $\{\hat{f}_{p,t}\}$ and $\{\bar{w}_{p,t}\}$ generated by Gradraker satisfy the following bound

$$\sum_{t=1}^{T} \mathcal{L}_t\left(\sum_{p=1}^{P} \bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{a}_t)\right) - \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_p^*(\mathbf{a}_t))$$
$$\leq \frac{\ln P}{\eta} + \frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta} + \frac{\eta L^2 T}{2} + \eta T \tag{26}$$

where $\boldsymbol{\theta}_p^*$ is associated with the best RF function approximant $\hat{f}_p^*(\mathbf{a}) = (\boldsymbol{\theta}_p^*)^\top \mathbf{z}_p(\mathbf{a})$.

Proof: See Appendix A ∎

In addition to bounding the regret in the RF space, the next theorem compares the Gradraker loss relative to that of the best functional estimator in the original RKHS.

Theorem 2: Under (as1)–(as3), and with f^* defined as in (25), for a fixed $\epsilon > 0$, the following bound holds with probability at least $1 - 2^8(\frac{\sigma_p}{\epsilon})^2 \exp(\frac{-D\epsilon^2}{4N+8})$

$$\sum_{t=1}^{T} \mathcal{L}_t\left(\sum_{p=1}^{P} \bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{a}_t)\right) - \sum_{t=1}^{T} \mathcal{L}_t\left(f^*(\mathbf{a}_t)\right)$$
$$\leq \frac{\ln P}{\eta} + \frac{(1+\epsilon)C^2}{2\eta} + \frac{\eta L^2 T}{2} + \eta T + \epsilon LTC \tag{27}$$

where C is a constant, while $\sigma_p^2 := \mathbb{E}_{\pi_{\kappa_p}}[\|\mathbf{v}\|^2]$ is the second-order moment of the RF vector norm. Setting $\eta = \epsilon = \mathcal{O}(1/\sqrt{T})$ in (27), the static regret in (24) leads to

$$\mathrm{Reg}^{\mathrm{s}}_{\mathrm{Gradraker}}(T) = \mathcal{O}(\sqrt{T}). \tag{28}$$

Proof: See Appendix B ∎

Observe that the probability of (27) to hold grows as D increases, and one can always find a D to ensure a positive probability for a given ϵ. Theorem 2 establishes that with a proper choice of parameters, the Gradraker achieves sub-linear regret relative to the best static function approximant in (25), which means the novel Gradraker algorithm is capable of capturing the nonlinear relationship among nodal functions accurately, as long as enough nodes are sampled sequentially.

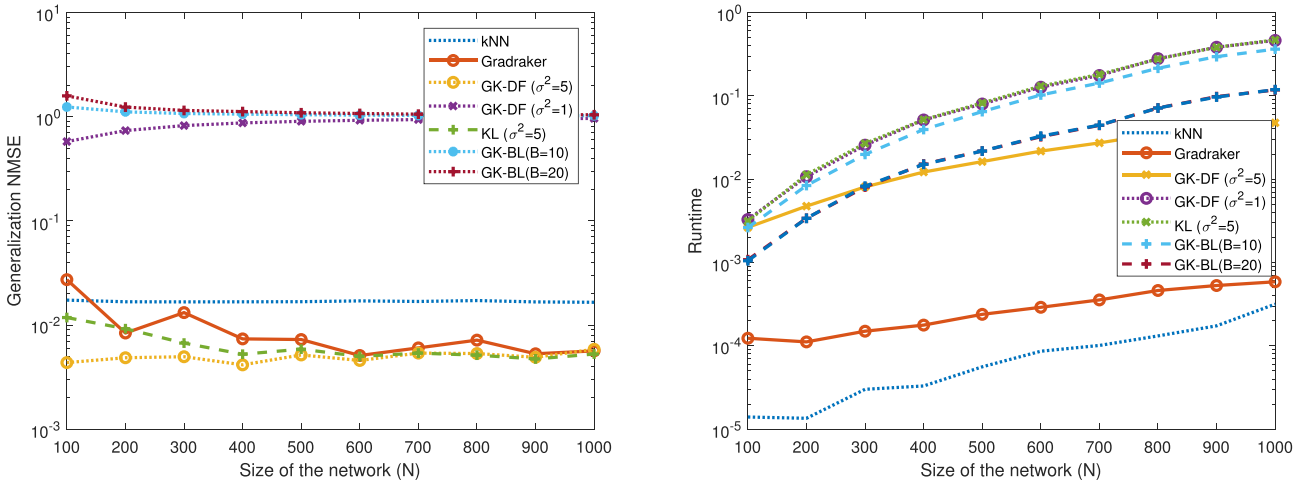
In addition, it is worth noting that Theorem 2 holds true regardless of the sampling order of the nodes $\{v_1, \ldots, v_T\}$. However, optimizing over the sampling pattern is possible, and constitutes one of our future research directions.

VI. Numerical Tests

In this section, Gradraker is tested on both synthetic and real datasets to corroborate its effectiveness. The tests will mainly focus on regression tasks for a fair comparison with existing alternatives.

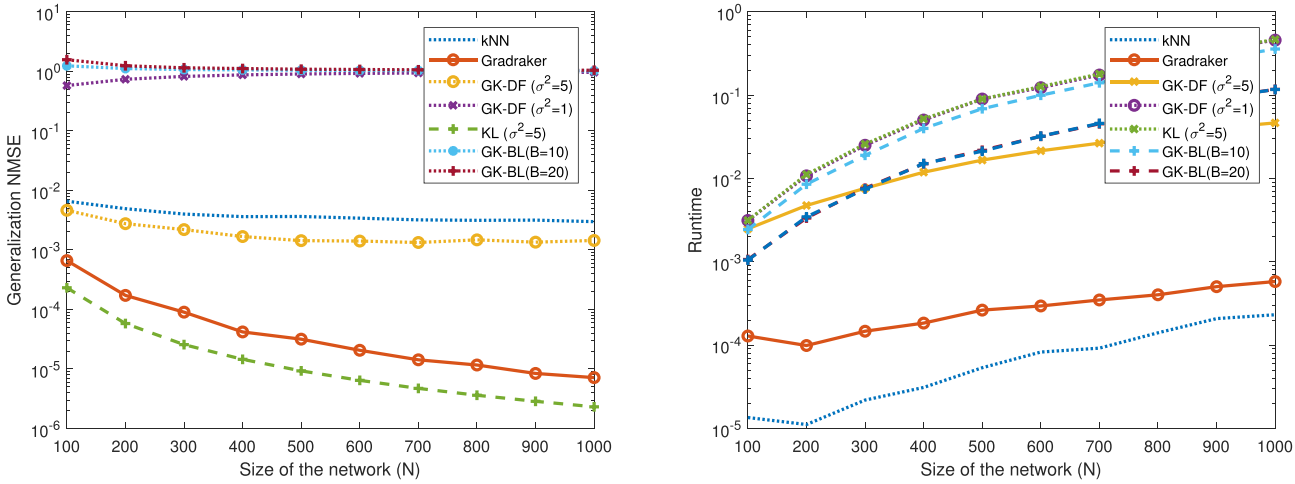
A. Synthetic Data Test

Data generation. An Erdös-Rényi graph [38] with binary adjacency matrix $\mathbf{A}_0 \in \mathbf{R}^{N\times N}$ was generated with probability of edge presence $\pi = 0.2$, and its adjacency was symmetrized as $\mathbf{A} = \mathbf{A}_0 + \mathbf{A}_0^\top$. This symmetrization is not required by Gradraker, but it is necessary for alternative graph kernel based methods. A function over this graph was then generated with each entry of the coefficient vector $\alpha \in \mathbb{R}^N$ drawn uniformly from [0.5, 1], and each entry of the noise $\mathbf{e}$ drawn from $\mathcal{N}(0, 0.01\mathbf{I})$. In each experiment, the sampling matrix $\boldsymbol{\Psi}$ is randomly generated so that $M = 0.05N$ of the nodes are randomly sampled, and the remaining $N - M$ nodes are treated as newly-joining nodes, whose function values and connectivity patterns are both unknown at the training phase, and whose nodal function values are estimated based on their connectivity with existing nodes in the network during the testing phase. All algorithms are carried out on the training set of M nodes, and the obtained model is used to estimate the function value on the newly arriving nodes. The runtime for estimating the



(a) Generalization NMSE (b) Testing runtime

Fig. 1. Inference performance versus number of nodes for synthetic dataset generated from graph diffusion kernel.



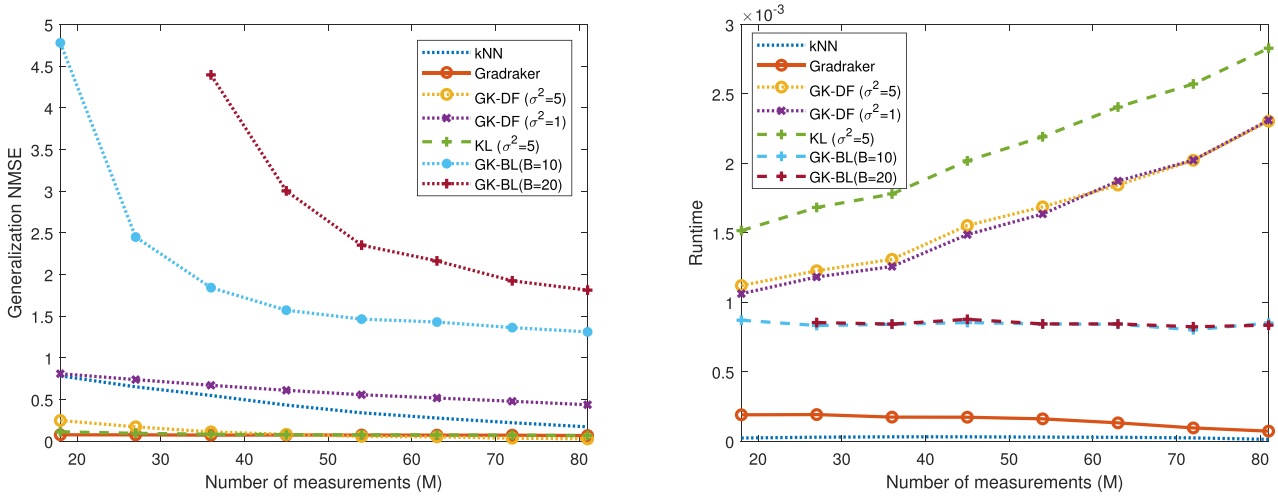
(a) Generalization NMSE (b) Runtime

Fig. 2. Inference performance versus number of nodes for synthetic dataset generated from Gaussian kernel.

function value on the newly-joining nodes, as well as the generalization NMSE $:= \frac{1}{|\mathcal{S}^c|}\|\hat{\mathbf{x}}_{\mathcal{S}^c} - \mathbf{x}_{\mathcal{S}^c}\|_2^2/\|\mathbf{x}_{\mathcal{S}^c}\|_2^2$ performance is evaluated, with $\mathcal{S}^c$ denoting the index set of new nodes. The Gradraker adopts a dictionary consisting of 2 Gaussian kernels with parameters $\sigma^2 = 1, 5$, using $D = 10$ random features, and it is compared with: a) the k NN algorithm, with k selected as the maximum number of neighbors a node has in a specific network, and with the combining weights set to $1/k$ in unweighted graphs, and $a_{il}/\sum_{j\in\mathcal{N}_i} a_{ij}$ for the lth neighbor in weighted graphs; b) the graph kernel (GK) based method using diffusion kernels with different bandwidths (named as GK-DF), or band-limited kernels with different bandwidths (GK-BL); and c) kernel based learning without RF approximation (KL) with a Gaussian kernel of $\sigma^2 = 5$. Results are averaged over 100 independent runs. The regularization parameter for all algorithms is selected from the set $\mu = \{10^{-7}, 10^{-6}, \ldots, 10^{0}\}$ via cross validation.

Testing results. Fig. 1 illustrates the performance in terms of the average runtime and NMSE versus the number of nodes (size) of the network. In this experiment, $\bar{\mathbf{K}}$ in (7) is generated from the normalized graph Laplacian $\mathbf{L}$, using the diffusion kernel $r(\lambda) = \exp(\sigma^2\lambda/2)$. A bandwidth of $\sigma^2 = 5$ was used to generate the data. It is observed that GK attains the best generalization accuracy when the ground-truth model is known, but its computational complexity grows rapidly with the network size. However, GK does not perform as well when a mismatched kernel is applied. The Gradraker method on the other hand, is very efficient, while at the same time it can provide reasonable estimates of the signal on the newly arriving nodes, even without knowledge about the kernels. The k-NN method is very efficient, but does not provide as reliable performance as the Gradraker.

Figure 2 depicts the performance of competitive algorithms. Matrix $\bar{\mathbf{K}}$ for data generation is formed based on (8) using the Gaussian kernel $\kappa(\mathbf{a}_i - \mathbf{a}_j) = \exp(\|\mathbf{a}_i - \mathbf{a}_j\|^2/\sigma^2)$, with $\sigma^2 = 5$. In this case, KL exactly matches the true model, and hence it achieves the best performance. However, it is the most complex in terms of runtime. Meanwhile, GK-based methods



(a) Generalization NMSE (b) Runtime

Fig. 3. Inference performance versus number of sampled nodes in temperature dataset.

suffer from model mismatch, and are also relatively more complex than Graderaker. The novel Gradraker is capable of estimating the nodal function on the newly joining nodes with high accuracy at very low computational complexity. Note that in real-world scenarios, accurate prior information about the underlying model is often unavailable, in which case Gradraker can be a more reliable and efficient choice.

B. Reconstruction of the Temperature Data

This subsection tests the performance of Gradraker on a real temperature dataset. The dataset comprises 24 signals corresponding to the average temperature per month in the intervals $1961-1980$ and $1991-2010$ measured by $N_a = 89$ stations in Switzerland [39]. The training set contains the first 12 signals, corresponding to the interval $1961-1980$, while the test set contains the remaining 12. Each station is represented by a node, and the graph was constructed using the algorithm in [40] based on the training signals. Given the test signal on a randomly chosen subset of $N = M$ vertices, the values at the remaining $N_a - M$ vertices are estimated as newly-coming nodes. The generalization NMSE over the $N_a - M$ nodes is averaged across the test signals.

Fig. 3 compares the performance of Gradraker with those of competing alternatives. Gradraker adopts a dictionary consisting of 3 Gaussian kernels with parameters $\sigma^2 = 1, 5, 10$, using $D = 100$ random features. It is clear from Fig. 3 that Gradraker outperforms GK in both generalization NMSE and runtime. On the other hand, even though KL achieves lower generalization NMSE, it incurs a much higher complexity.

C. Reconstruction of the Email-Eu-Core Data

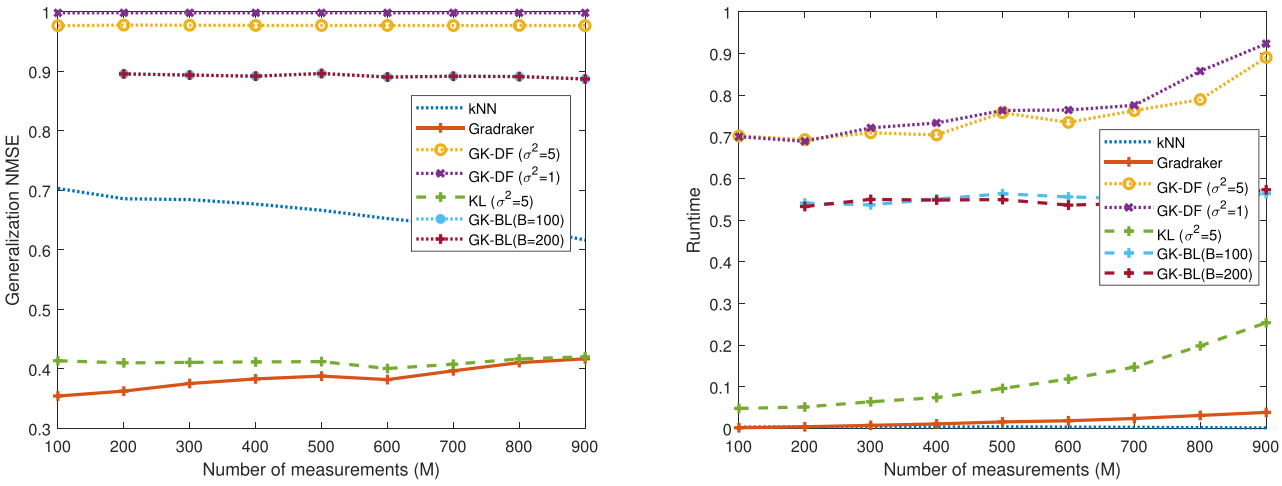
The Eu-core network was generated using email data from a large European research institution [41], where each node represents a person, and an edge (i, j) is present if person i sent person j at least one email. The e-mails only represent communication between institution members (the core), and the dataset does not contain incoming messages from or outgoing messages to the rest of the world. The dataset also contains "ground-truth" community memberships of the nodes. Each individual belongs to one of 42 departments at the research institute. During the experiment, the department labels are considered to be y_n that are to be sampled and estimated. The graph consists of $1,005$ nodes, and 25,571 edges. Gradraker adopts a dictionary consisting of 2 Gaussian kernels with parameters $\sigma^2 = 1, 10$, from which $D = 10$ random features are generated. The test results were averaged over 100 independent runs with randomly sampled nodes.

Fig. 4 compares the performance of Gradraker with those of alternative algorithms when different numbers of nodal labels are observed. It is clear that the RF-based approach outperforms the GK-based method in both reconstruction accuracy and runtime. While the batch KL method without RF approximation outperforms the RF method, it incurs considerably higher computational complexity.

D. Reconstruction of the Cora Data

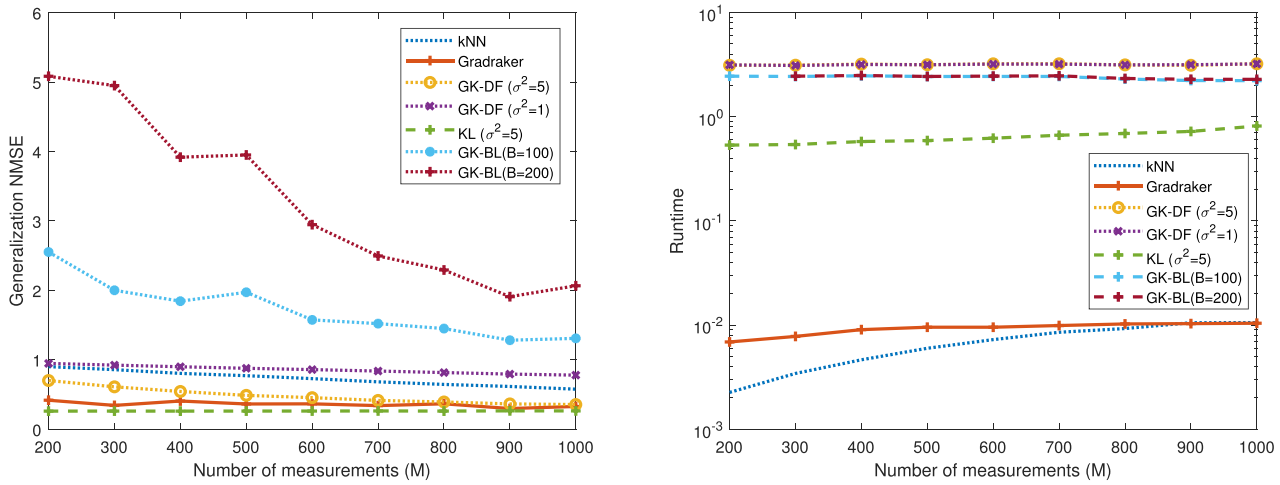
This subsection tests the Gradraker algorithm on the Cora citation dataset [5]. Gradraker adopts a dictionary consisting of 2 Gaussian kernels with parameters $\sigma^2 = 1, 10$, using $D = 20$ random features. The results were averaged over 100 independent runs. The Cora dataset consists of 2,708 scientific publications classified into one of seven classes. The citation network consists of 5,429 links. The network is constructed so that a link connects node i to node j if paper i cites paper j, and the category id the paper belongs to is to be reconstructed. It can be observed again from Fig. 5, that the Gradraker markedly outperforms the GK algorithms in terms of generalization NMSE, and is much more computationally efficient than all other algorithms except the kNN method, which however does not perform as well.

It can be readily observed from our numerical results over synthetic and real datasets, that the Gradraker provides reliable performance in terms of NMSE in all tests, while at the same



(a) Generalization NMSE (b) Runtime

Fig. 4. Inference performance versus number of sampled nodes in email dataset.



(a) Generalization NMSE (b) Runtime

Fig. 5. Inference performance versus number of sampled nodes in Cora dataset.

time, it scales much better than all kernel based alternatives. This is because the alternative kernel-based algorithms require re-computing the kernel matrix whenever a new node joins the network. It is worth noting that all kernel-based alternatives require exact knowledge of the entire network topology, which is not necessary for GradRaker that only requires $\{\mathbf{z}_{\mathbf{V}}(\mathbf{a}_n)\}$. These tests corroborate the potential of GradRaker for application settings, where the graphs grow and nodes have privacy constraints.

VII. Conclusions

The present paper deals with the problem of reconstructing signals over graphs, from samples over a subset of nodes. An online MKL based algorithm is developed, which is capable of estimating and updating the nodal functions even when samples are collected sequentially. The novel online scheme is highly scalable and can estimate the unknown signals on newly joining nodes. Unlike many existing approaches, it only relies on encrypted nodal connectivity information, which is appealing for networks where nodes have strict privacy constraints.

This work opens up a number of interesting directions for future research, including: a) exploring distributed implementations that are well motivated in large-scale networks; b) graph-adaptive learning when multiple sets of features are available; and c) developing adaptive sampling strategies for Gradraker.

Appendix A
Proof of Lemma 1

To prove Lemma 1, we introduce two intermediate lemmata.

Lemma 3: Under (as1), (as2), and $\hat{f}_p^*$ as in (25) with $\mathcal{F}_p := \{\hat{f}_p | \hat{f}_p(\mathbf{a}) = \boldsymbol{\theta}^\top \mathbf{z}_p(\mathbf{a}), \forall \boldsymbol{\theta} \in \mathbb{R}^{2D}\}$, let $\{\hat{f}_{p,t}(\mathbf{a}_t)\}$ denote the sequence of estimates generated by Gradraker with a pre-selected kernel κ_p. Then the following bound holds true w.p.1

$$\sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_{p,t}(\mathbf{a}_t)) - \sum_{t=1}^{T} \mathcal{L}_t(\hat{f}_p^*(\mathbf{a}_t)) \leq \frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta} + \frac{\eta L^2 T}{2} \quad (29)$$

where η is the learning rate, L is the Lipschitz constant in (as2), and $\boldsymbol{\theta}_p^*$ is the corresponding parameter (or weight) vector supporting the best estimator $\hat{f}_p^*(\mathbf{a}) = (\boldsymbol{\theta}_p^*)^\top \mathbf{z}_p(\mathbf{a})$.

Proof: The proof is similar to the regret analysis of online gradient descent, see e.g., [29]. ■

In addition, we will bound the difference between the loss of the solution obtained from Algorithm 2 and the loss of the best single kernel-based online learning algorithm. Specifically, the following lemma holds.

Lemma 4: Under (as1) and (as2), with $\{\hat{f}_{p,t}\}$ generated from Gradraker, it holds that

$$\sum_{t=1}^{T}\sum_{p=1}^{P} \bar{w}_{p,t}\mathcal{L}_t(\hat{f}_{p,t}(\mathbf{a}_t)) - \sum_{t=1}^{T}\mathcal{L}_t(\hat{f}_{p,t}(\mathbf{a}_t)) \leq \eta T + \frac{\ln P}{\eta} \tag{30}$$

where η is the learning rate in (22), and P is the number of kernels in the dictionary.

Proof: Letting $W_t := \sum_{p=1}^{P} w_{p,t}$, the weight recursion in (22) implies that

$$\begin{aligned} W_{t+1} &= \sum_{p=1}^{P} w_{p,t+1} = \sum_{p=1}^{P} w_{p,t}\exp\left(-\eta\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)\right) \\ &\leq \sum_{p=1}^{P} w_{p,t}\left(1-\eta\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)+\eta^2\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)^2\right) \end{aligned} \tag{31}$$

where the last inequality holds because $\exp(-\eta x) \leq 1 - \eta x + \eta^2 x^2$, for $|\eta| \leq 1$. Furthermore, substituting $\bar{w}_{p,t} := w_{p,t}/\sum_{p=1}^{P} w_{p,t} = w_{p,t}/W_t$ into (31) leads to

$$\begin{aligned} W_{t+1} &\leq \sum_{p=1}^{P} W_t\bar{w}_{p,t}\left(1-\eta\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)+\eta^2\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)^2\right) \\ &= W_t\left(1-\eta\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)\right. \\ &\qquad \left.+\eta^2\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)^2\right). \end{aligned} \tag{32}$$

Since $1 + x \leq e^x$, $\forall x$, it follows that

$$\begin{aligned} W_{t+1} \leq W_t\exp\left(-\eta\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)\right. \\ \left.+\eta^2\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)^2\right). \end{aligned} \tag{33}$$

Telescoping (33) from $t = 1$ to T yields

$$\begin{aligned} W_{T+1} \leq \exp\left(-\eta\sum_{t=1}^{T}\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)\right. \\ \left.+\eta^2\sum_{t=1}^{T}\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)^2\right). \end{aligned} \tag{34}$$

On the other hand, for any p, it holds that

$$\begin{aligned} W_{T+1} &\geq w_{p,T+1} \\ &= w_{p,1}\prod_{t=1}^{T}\exp\left(-\eta\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)\right) \\ &= w_{p,1}\exp\left(-\eta\sum_{t=1}^{T}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)\right). \end{aligned} \tag{35}$$

Combining (34) with (35), we arrive at

$$\begin{aligned} &\exp\left(-\eta\sum_{t=1}^{T}\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)\right. \\ &\qquad \left.+\eta^2\sum_{t=1}^{T}\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)^2\right) \\ &\geq w_{p,1}\exp\left(-\eta\sum_{t=1}^{T}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)\right). \end{aligned} \tag{36}$$

Taking the logarithm on both sides of (36), and recalling that $w_{p,1} = 1/P$, we obtain

$$\begin{aligned} &-\eta\sum_{t=1}^{T}\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)+\eta^2\sum_{t=1}^{T}\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)^2 \\ &\geq -\eta\sum_{t=1}^{T}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right) - \ln P. \end{aligned} \tag{37}$$

Re-organizing the terms leads to

$$\begin{aligned} &\sum_{t=1}^{T}\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right) \\ &\leq \sum_{t=1}^{T}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)+\eta\sum_{t=1}^{T}\sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)^2+\frac{\ln P}{\eta} \end{aligned} \tag{38}$$

and the proof is complete, since $\mathcal{L}_t(\hat{f}_{p,t}(\mathbf{a}_t))^2 \leq 1$ and $\sum_{p=1}^{P}\bar{w}_{p,t} = 1$. ■

Since $\mathcal{L}_t(\cdot)$ is convex under (as1), Jensen's inequality implies

$$\mathcal{L}_t\left(\sum_{p=1}^{P}\bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{a}_t)\right) \leq \sum_{p=1}^{P}\bar{w}_{p,t}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right). \tag{39}$$

Combining (39) with Lemma 4, one arrives readily at

$$\begin{aligned} &\sum_{t=1}^{T}\mathcal{L}_t\left(\sum_{p=1}^{P}\bar{w}_{p,t}\hat{f}_{p,t}(\mathbf{a}_t)\right) \\ &\leq \sum_{t=1}^{T}\mathcal{L}_t\left(\hat{f}_{p,t}(\mathbf{a}_t)\right)+\eta T+\frac{\ln P}{\eta} \\ &\overset{(a)}{\leq} \sum_{t=1}^{T}\mathcal{L}_t\left(\hat{f}_p^*(\mathbf{a}_t)\right)+\frac{\ln P}{\eta}+\frac{\|\boldsymbol{\theta}_p^*\|^2}{2\eta}+\frac{\eta L^2 T}{2}+\eta T \end{aligned} \tag{40}$$

where (a) follows due to Lemma 3 and because $\boldsymbol{\theta}_p^*$ is the optimal solution for any given κ_p. This proves Lemma 1.

APPENDIX B
PROOF OF THEOREM 2

To bound the performance relative to the best estimator $f^*(\mathbf{a}_t)$ in the RKHS, the key step is to bound the approximation error. For a given shift-invariant κ_p, the maximum point-wise error of the RF kernel approximant is bounded with probability at least $1-2^8(\frac{\sigma_p}{\epsilon})^2\exp(\frac{-D\epsilon^2}{4N+8})$, by [25]

$$\sup_{\mathbf{a}_i,\mathbf{a}_j\in\mathcal{X}}\left|\mathbf{z}_p^\top(\mathbf{a}_i)\mathbf{z}_p(\mathbf{a}_j)-\kappa_p(\mathbf{a}_i,\mathbf{a}_j)\right|<\epsilon \tag{41}$$

where $\epsilon>0$ is a given constant, D the number of features, while M is the number of nodes already in the network, and $\sigma_p^2:=\mathbb{E}_p[\|\mathbf{v}\|^2]$ is the second-order moment of the RF vector norm induced by κ_p. Henceforth, for the optimal function estimator (25) in $\mathcal{H}_p$ denoted by $f_p^*(\mathbf{a}):=\sum_{t=1}^T\alpha_{p,t}^*\kappa_p(\mathbf{a},\mathbf{a}_t)$, and its RF-based approximant $\check{f}_p^*:=\sum_{t=1}^T\alpha_{p,t}^*\mathbf{z}_p^\top(\mathbf{a})\mathbf{z}_p(\mathbf{a}_t)\in\mathcal{F}_p$, we have

$$\begin{aligned}
&\left|\sum_{t=1}^T\mathcal{L}_t\left(\check{f}_p^*(\mathbf{a}_t)\right)-\sum_{t=1}^T\mathcal{L}_t\left(f_p^*(\mathbf{a}_t)\right)\right|\\
&\overset{(a)}{\leq}\sum_{t=1}^T\left|\mathcal{L}_t\left(\check{f}_p^*(\mathbf{a}_t)\right)-\mathcal{L}_t(f_p^*(\mathbf{a}_t))\right|\\
&\overset{(b)}{\leq}\sum_{t=1}^T L\left|\sum_{t'=1}^T\alpha_{p,t'}^*\mathbf{z}_p^\top(\mathbf{a}_{t'})\mathbf{z}_p(\mathbf{a}_t)-\sum_{t'=1}^T\alpha_{p,t'}^*\kappa_p(\mathbf{a}_{t'},\mathbf{a}_t)\right|\\
&\overset{(c)}{\leq}\sum_{t=1}^T L\sum_{t'=1}^T|\alpha_{p,t'}^*|\left|\mathbf{z}_p^\top(\mathbf{a}_{t'})\mathbf{z}_p(\mathbf{a}_t)-\kappa_p(\mathbf{a}_{t'},\mathbf{a}_t)\right|
\end{aligned}\tag{42}$$

where (a) is due to the triangle inequality; (b) uses the Lipschitz continuity of the loss, and (c) is due to the Cauchy-Schwarz inequality. Combining with (41), yields

$$\begin{aligned}
&\left|\sum_{t=1}^T\mathcal{L}_t(\check{f}_p^*(\mathbf{a}_t))-\sum_{t=1}^T\mathcal{L}_t(f_p^*(\mathbf{a}_t))\right|\\
&\qquad\leq\sum_{t=1}^T L\epsilon\sum_{t'=1}^T|\alpha_{p,t'}^*|\leq\epsilon LTC,\ \text{w.h.p.}
\end{aligned}\tag{43}$$

where we used that $C:=\max_p\sum_{t=1}^T|\alpha_{p,t}^*|$. Under the kernel bounds in (as3), the uniform convergence in (41) implies that $\sup_{\mathbf{a}_t,\mathbf{a}_{t'}\in\mathcal{X}}\mathbf{z}_p^\top(\mathbf{a}_t)\mathbf{z}_p(\mathbf{a}_{t'})\leq 1+\epsilon$, w.h.p., which leads to

$$\begin{aligned}
\left\|\boldsymbol{\theta}_p^*\right\|^2&:=\left\|\sum_{t=1}^T\alpha_{p,t}^*\mathbf{z}_p(\mathbf{a}_t)\right\|^2\\
&=\left|\sum_{t=1}^T\sum_{t'=1}^T\alpha_{p,t}^*\alpha_{p,t'}^*\mathbf{z}_p^\top(\mathbf{a}_t)\mathbf{z}_p(\mathbf{a}_{t'})\right|\leq(1+\epsilon)C^2
\end{aligned}\tag{44}$$

where for the last inequality we used the definition of C.

Lemma 1 together with (43) and (44) lead to the regret of the proposed Gradraker algorithm relative to the best static function in $\mathcal{H}_p$, that is given by

$$\begin{aligned}
&\sum_{t=1}^T\mathcal{L}_t\left(\sum_{p=1}^P w_{p,t}\hat{f}_{p,t}(\mathbf{a}_t)\right)-\sum_{t=1}^T\mathcal{L}_t(f_p^*(\mathbf{a}_t))\\
&=\sum_{t=1}^T\mathcal{L}_t\left(\sum_{p=1}^P w_{p,t}\hat{f}_{p,t}(\mathbf{a}_t)\right)-\sum_{t=1}^T\mathcal{L}_t\left(\check{f}_p^*(\mathbf{a}_t)\right)\\
&\quad+\sum_{t=1}^T\mathcal{L}_t\left(\check{f}_p^*(\mathbf{a}_t)\right)-\sum_{t=1}^T\mathcal{L}_t(f_p^*(\mathbf{a}_t))\\
&\leq\frac{\ln P}{\eta}+\frac{\eta L^2T}{2}+\eta T+\frac{(1+\epsilon)C^2}{2\eta}+\epsilon LTC,\ \text{w.h.p.}
\end{aligned}\tag{45}$$

which completes the proof of Theorem 2.

REFERENCES

[1] E. D. Kolaczyk, *Statistical Analysis of Network Data: Methods and Models*. Berlin, Germany: Springer, 2009.

[2] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," in *Proc. Int. Conf. Mach. Learn.*, Sydney, Australia, Jul. 2002, pp. 315–322.

[3] 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, pp. 2399–2434, Nov. 2006.

[4] L. Wasserman and J. D. Lafferty, "Statistical analysis of semi-supervised regression," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, Canada, 2008, pp. 801–808.

[5] Q. Lu and L. Getoor, "Link-based classification," in *Proc. Int. Conf. Mach. Learn.*, Washington, DC, USA, 2003, pp. 496–503.

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

[7] O. Chapelle, B. Scholkopf, and A. Zien, "Semi-supervised learning," *IEEE Trans. Neural Netw.*, vol. 20, no. 3, pp. 542–542, Mar. 2009.

[8] O. Chapelle, V. Vapnik, and J. Weston, "Transductive inference for estimating values of functions," in *Proc. Adv. Neural Inf. Process. Syst.*, Denver, CO, USA, 1999, pp. 421–427.

[9] C. Cortes and M. Mohri, "On transductive regression," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, Canada, Dec. 2006, pp. 305–312.

[10] D. Berberidis, A. N. Nikolakopoulos, and G. B. Giannakis, "Adaptive diffusions for scalable learning over graphs," 2018, arXiv:1804.02081.

[11] S. K. Narang, A. Gadde, and A. Ortega, "Signal processing techniques for interpolation in graph structured data," in *Proc IEEE Int. Conf. Acoust., Speech, Signal Process.*, Vancouver, Canada, 2013, pp. 5445–5449.

[12] X. Wang, P. Liu, and Y. Gu, "Local-set-based graph signal reconstruction," *IEEE Trans. Signal Process.*, vol. 63, no. 9, pp. 2432–2444, May 2015.

[13] D. Romero, M. Ma, and G. B. Giannakis, "Kernel-based reconstruction of graph signals," *IEEE Trans. Signal Process.*, vol. 65, no. 3, pp. 764–778, Feb. 2017.

[14] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro, "Sampling of graph signals with successive local aggregations," *IEEE Trans. Signal Process.*, vol. 64, no. 7, pp. 1832–1843, Apr. 2016.

[15] 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.

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

[17] V. N. Ioannidis, Y. Shen, and G. B. Giannakis, "Semi-blind inference of topologies and dynamical processes over graphs," 2018, arXiv:1805.06095.

[18] G. Wahba, *Spline Models for Observational Data*. Philadelphia, PA, USA: SIAM, 1990.

[19] P. Di Lorenzo, P. Banelli, E. Isufi, S. Barbarossa, and G. Leus, "Adaptive graph signal processing: Algorithms and optimal sampling strategies," *IEEE Trans. Signal Process.*, vol. 66, no. 13, pp. 3584–3598, Jul. 2018.

[20] V. N. Ioannidis, D. Romero, and G. B. Giannakis, "Inference of spatio-temporal functions over graphs via multikernel kriged Kalman filtering," *IEEE Trans. Signal Process.*, vol. 66, no. 12, pp. 3228–3239, Jun. 2018.

[21] A. Grover and J. Leskovec, "node2vec: Scalable feature learning for networks," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 855–864.
[22] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2014, pp. 701–710.
[23] N. S. Altman, "An introduction to kernel and nearest-neighbor nonparametric regression," *Amer. Statistician*, vol. 46, no. 3, pp. 175–185, Feb. 1992.
[24] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," in *Proc. Adv. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 1024–1034.
[25] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, Dec. 2007, pp. 1177–1184.
[26] Y. Shen, T. Chen, and G. B. Giannakis, "Online ensemble multi-kernel learning adaptive to non-stationary and adversarial environments," in *Proc. Int. Conf. Artif. Intell. Statist.*, Lanzarote, Canary Islands, Apr. 2018.
[27] E. Hazan, "Introduction to online convex optimization," *Found. Trends Mach. Learn.*, vol. 2, no. 3/4, pp. 157–325, 2016.
[28] C. Cortes, M. Mohri, and A. Rostamizadeh, "ℓ_2-regularization for learning kernels," in *Proc. Conf. Uncertainty Artif. Intell.*, Montreal, ON, Canada, Jun. 2009, pp. 109–116.
[29] Y. Shen, T. Chen, and G. B. Giannakis, "Random feature-based online multi-kernel learning in environments with unknown dynamics," *J. Mach. Learn. Res.*, vol. 20, no. 22, pp. 1–36, 2019.
[30] J. Chen, H. Fang, and Y. Saad, "Fast approximate kNN graph construction for high dimensional data via recursive Lanczos bisection," *J. Mach. Learn. Res.*, vol. 10, pp. 1989–2012, Sep. 2009.
[31] M. Kivelä, A. Arenas, M. Barthelemy, J. P. Gleeson, Y. Moreno, and M. A. Porter, "Multilayer networks," *J. Complex Netw.*, vol. 2, no. 3, pp. 203–271, 2014.
[32] P. Traganitis, Y. Shen, and G. B. Giannakis, "Topology inference for multilayer networks," in *Proc. Int. Workshop Netw. Sci. Commun.*, Atlanta, GA, May 2017, pp. 898–903.
[33] D. Zhou, S. A. Orshanskiy, H. Zha, and C. L. Giles, "Co-ranking authors and documents in a heterogeneous network," in *Proc. Int. Conf. Data Mining*, Omaha, NE, USA, Oct. 2007, pp. 739–744.
[34] Y. Sun and J. Han, "Mining heterogeneous information networks: A structural analysis approach," *ACM SIGKDD Explorations Newslett.*, vol. 14, no. 2, pp. 20–28, 2013.
[35] S. Shalev-Shwartz, "Online learning and online convex optimization," *Found. Trends Mach. Learn.*, vol. 4, no. 2, pp. 107–194, 2011.
[36] C. A. Micchelli and M. Pontil, "Learning the kernel function via regularization," *J. Mach. Learn. Res.*, vol. 6, pp. 1099–1125, Jul. 2005.
[37] J. Lu, S. C. Hoi, J. Wang, P. Zhao, and Z.-Y. Liu, "Large scale online kernel learning," *J. Mach. Learn. Res.*, vol. 17, no. 47, pp. 1–43, Apr. 2016.
[38] P. Erdos and A. Rényi, "On random graphs I," *Publ. Math. Debrecen*, vol. 6, pp. 290–297, 1959.
[39] "Meteorology and climatology meteoswiss." 2014. [Online]. Available: http://www.meteoswiss.admin.ch/home/climate/past/climate-normals/climate-diagrams-and-normal-values-per-station.html
[40] 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.
[41] J. Leskovec, J. Kleinberg, and C. Faloutsos, "Graph evolution: Densification and shrinking diameters," *ACM Trans. Knowl. Discov. Data*, vol. 1, no. 1, Mar. 2007, Art. no. 2.

Yanning Shen (S'13) received the B.Sc. and M.Sc. degrees in electrical engineering from the University of Electronic Science and Technology of China, Chengdu, China, in 2011 and 2014, respectively. Since September 2014, she has been working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, University of Minnesota (UMN), Minneapolis, MN, USA. Her research interests include data science, network science, machine learning, optimization, and statistical signal processing. She received the UESTC distinguished B.Sc. thesis Award in 2011 and distinguished M.Sc. Thesis Award in 2014. She was a Best Student Paper Award finalist of the 2017 IEEE International Workshop on Computational Advances in Multi-Sensor Adaptive Processing and the 2017 Asilomar Conference on Signals, Systems, and Computers. She was selected as Rising Stars in EECS by Stanford University in 2017, and received the UMN Doctoral Dissertation Fellowship in 2018.

Geert Leus received the M.Sc. and Ph.D. degrees in electrical engineering from the KU Leuven, Leuven, Belgium, in June 1996 and May 2000, respectively. He is currently an "Antoni van Leeuwenhoek" Full Professor with the Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands. His research interests are in the broad area of signal processing, with a specific focus on wireless communications, array processing, sensor networks, and graph signal processing. He received a 2002 IEEE Signal Processing Society Young Author Best Paper Award and a 2005 IEEE Signal Processing Society Best Paper Award. He is a Fellow of EURASIP, a Member-at-Large of the Board of Governors of the IEEE Signal Processing Society, the Chair of the IEEE Signal Processing for Communications and Networking Technical Committee, a member of the IEEE Sensor Array and Multichannel Technical Committee, and the Editor in Chief of the *EURASIP Journal on Advances in Signal Processing*. He was also on the Editorial Boards of the IEEE TRANSACTIONS ON SIGNAL PROCESSING, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS, IEEE SIGNAL PROCESSING LETTERS, and the EURASIP JOURNAL ON ADVANCES IN SIGNAL PROCESSING. He is currently the Chair of the EURASIP Special Area Team on Signal Processing for Multisensor Systems, a member of the IEEE Signal Processing Theory and Methods Technical Committee, a member of the IEEE Big Data Special Interest Group, an Associate Editor of *Foundations and Trends in Signal Processing*, and the Editor in Chief of *EURASIP Signal Processing*.

Georgios B. Giannakis (F'97) received the Diploma in electrical engineering from the National Technical University of Athens, Athens, Greece, in 1981, and the M.Sc. and Ph.D. degrees in electrical engineering from the University of Southern California (USC), Los Angeles, CA, USA, in 1983 and 1986. From 1982 to 1986, he was with the USC. He was a faculty member with the University of Virginia from 1987 to 1998, and since 1999, he has been a professor with the University of Minnesota, Minneapolis, MN, USA, where he holds an ADC Endowed Chair, a University of Minnesota McKnight Presidential Chair in ECE, and is the Director of the Digital Technology Center. His general interests span the areas of statistical learning, communications, and networking—subjects on which he has authored or co-authored more than 450 journal papers, 750 conference papers, 25 book chapters, two edited books and two research monographs (h-index 140). His current research focuses on data science, and network science with applications to the Internet of Things, social, brain, and power networks with renewables. He is the (co-)inventor of 32 patents issued, and the (co-)recipient of nine best journal paper awards from the IEEE Signal Processing (SP) and Communications Societies, including the G. Marconi Prize Paper Award in Wireless Communications. He also received Technical Achievement Awards from the SP Society (2000), from EURASIP (2005), a Young Faculty Teaching Award, the G. W. Taylor Award for Distinguished Research from the University of Minnesota, and the IEEE Fourier Technical Field Award (inaugural recipient in 2015). He is a Fellow of EURASIP, and has served the IEEE in a number of posts, including that of a Distinguished Lecturer for the IEEE-SP Society.