KERNEL REGRESSION ON GRAPHS IN RANDOM FOURIER FEATURES SPACE

Vitor R. M. Elias[⋆‡] *Vinay C. Gogineni*[§] *Wallace A. Martins*[†‡] *Stefan Werner*[⋆]

[⋆] Department of Electronic Systems, NTNU – Norwegian University of Science and Technology
[§]Department of Machine Intelligence, SimulaMet, Simula Research Laboratory
[†]Interdisciplinary Centre for Security, Reliability and Trust, UniLu – University of Luxembourg
[‡]Electrical Engineering Program, UFRJ – Federal University of Rio de Janeiro
Email: vitor.elias@ntnu.no, vcgogineni@simula.no, wallace.alvesmartins@uni.lu, stefan.werner@ntnu.no

ABSTRACT

This work proposes an efficient batch-based implementation for kernel regression on graphs (KRG) using random Fourier features (RFF) and a low-complexity online implementation. Kernel regression has proven to be an efficient learning tool in the graph signal processing framework. However, it suffers from poor scalability inherent to kernel methods. We employ RFF to overcome this issue and derive a batch-based KRG whose model size is independent of the training sample size. We then combine it with a stochastic gradient-descent approach to propose an online algorithm for KRG, namely the stochastic-gradient KRG (SGKRG). We also derive sufficient conditions for convergence in the mean sense of the online algorithms. We validate the performance of the proposed algorithms through numerical experiments using both synthesized and real data. Results show that the proposed batch-based implementation can match the performance of conventional KRG while having reduced complexity. Moreover, the online implementations effectively learn the target model and achieve competitive performance compared to the batch implementations.

***Index Terms*—** kernel regression on graphs, online learning on graphs, random Fourier features, stochastic gradient.

1. INTRODUCTION

The connectivity of real-world elements and the amount of data generated in networks have increased over the last decades [1, 2]. Real networks and their corresponding data come in vastly different shapes and applications, ranging from genetic interaction networks [3] and the human brain [4] to sensor networks and smart cities [5]. Although an extensive range of classical digital signal processing (DSP) tools are available, they are not directly applicable to information processing of signals from networked structures. The graph signal processing (GSP) emerged in the last decade as a suitable framework for signal processing over networks, leveraging the network structure to process the networked data [4–7].

A major area of research in GSP is learning over graphs, which aims at discovering patterns in the data and graph structure to allow, e.g., prediction and reconstruction of graph signals [10–18]. Among the several approaches for learning over graphs, kernel regression has proved to be an innovative technique in several estimation and reconstruction applications for networked data [14–17]. In this work, we build upon the methodology of kernel regression on graphs (KRG) proposed in [16], which embeds a metric of smoothness over the graph in the optimization problem to improve the learning of regression parameters. However, the methodology in [16] suffers from scalability issues from kernel methods and is restricted to a batch-based offline approach. In this work, we first derive an efficient batch-based KRG using random Fourier features [19]. Then, we propose an online strategy for KRG using the stochastic gradient descent approach.

This paper is organized as follows. Section 2 presents the basic concepts of graph signal processing, and formulates the problem of kernel regression over graphs. The proposed batch-based KRG using RFF is presented in Section 3, and in Section 4, we present the online strategy for KRG, namely, the stochastic-gradient KRG (SGKRG). In Section 5, we derive sufficient conditions for the update step size to guarantee the proposed online algorithm's convergence. In Section 7, we present the final remarks of this work.

2. LEARNING OVER GRAPHS

2.1. Graph Signal Processing

A graph is denoted by $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, where $\mathcal{V} = \{1, \ldots, K\}$ is the set of vertices, or nodes, and $\mathcal{E} = \{e_{11}, \ldots, e_{KK}\}$ is the set of edges. Elements $e_{ij} > 0$ indicate pairwise relations between nodes i and j according to a chosen metric, and an edge e_{ij} exists if and only if i and j are related [1,7]. In GSP, edges are typically represented in the adjacency matrix $\mathbf{A}$, such that the entry $A_{ij} = e_{ij}$ if e_{ij} exists and $A_{ij} = 0$ otherwise. In this work, we consider undirected graphs, such that $e_{ij} = e_{ji}$. The degree matrix $\mathbf{D}$ is a diagonal matrix such that $D_{jj} = \sum_{i \in \mathcal{N}_j} e_{ij}$ and $\mathcal{N}_j$ is the set of vertices that are adjacent to node j, referred to as neighborhood of j. The graph Laplacian is the positive semidefinite matrix $\mathbf{L} = \mathbf{D} - \mathbf{A}$ [2].

In GSP [2, 6], the signal on a graph is given by the mapping $s : \mathcal{V} \to \mathbb{R}$ and is represented by a vector $\mathbf{s}_n \in \mathbb{R}^K$. The graph signal represents a snapshot of the network state at time n. The graph Laplacian induces a variation metric for a graph signal $\mathbf{s}$ that depends on the graph structure [7]. The variation metric is given by

$$\nu(\mathbf{s}) = \mathbf{s}^{\mathrm{T}}\mathbf{L}\mathbf{s} = \sum_{i \neq j} A_{i,j}\big(s_i - s_j\big)^2, \tag{1}$$

where we can observe that the difference between two entries of the signal vector are penalized by the weight of the edge connecting the

The work of V. R. M. Elias was supported, in part, by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Grant number: 88887.310189/2018-00 and, in part, by CNPq. The work of Dr. Martins was supported, in part, by the ERC project AGNOSTIC and, in part, by FAPERJ. The work of Dr. Werner was supported, in part, by the Research Council of Norway.

two nodes. That is, a graph signal where values in adjacent nodes are different is associated with a large variation metric according to (1).

2.2. Learning Task

Consider a set of data pairs $\{\mathbf{x}_n, \mathbf{t}_n\}$, $n \in \{1, 2, \ldots, N\}$, such that vectors $\mathbf{x}_n$, called reference signals, are related to target signals $\mathbf{t}_n$ through an unknown function $f(\cdot)$ such that $\mathbf{t}_n = f(\mathbf{x}_n)$. The objective of typical learning strategies over graphs is to model $f(\cdot) : \mathbb{R}^K \rightarrow \mathbb{R}^K$ in the case where both $\mathbf{x}_n$ and $\mathbf{t}_n$ belong to the same graph $\mathcal{G}$. Previous research on learning over networks include, e.g., dictionary learning [11], linear [12, 13] and nonlinear graph filtering [18], kriged Kalman filtering [10], and kernel regression strategies [14–17].

In particular, the work in [16] proposes a kernel regression methodology that allows the reference signal to be agnostic to the graph. That is, the regression signal does not need to be a graph signal, whereas the target signal lies over $\mathcal{G}$, which widens the range of applications for the proposed method. For this, [16] assumes that graph signals are expected to be smooth with respect to the graph, inducing a low value of the variation metric (1). The model is then estimated in terms of a matrix $\mathbf{W}$ such that

$$\mathbf{y}_n = \mathbf{W}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}_n), \tag{2}$$

where $\mathbf{y}_n$ is an estimate of the target graph signal $\mathbf{t}_n$ and $\boldsymbol{\phi}(\cdot)$ is an unknown function of the input signal. The optimal parameter matrix $\mathbf{W}$ is found by minimizing the cost function

$$C(\mathbf{W}) = \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \mathrm{tr}(\mathbf{W}^{\mathrm{T}}\mathbf{W}) + \beta \sum_{n=1}^{N} \nu(\mathbf{y}_n), \tag{3}$$

where the last term on the right-hand side enforces that the model respects the smoothness of the target signal. The solution for (3) is obtained in [16] in closed form using the kernel method. This leads to a model whose dimension increases with the number of training samples. In the next sections, we propose a reduced-complexity solution for the batch-based solution of (3) using RFF, and we propose an online implementation for KRG.

3. BATCH-BASED KRG USING RANDOM FOURIER FEATURES

A way to overcome the scaling issues of kernel methods is provided by random Fourier features [19]. Using RFF, a shift-invariant kernel evaluation $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \kappa(\mathbf{x}_i - \mathbf{x}_j)$ is approximated as an inner product in the D-dimensional RFF space, where D is much lower than the number of training samples. The mapping of $\mathbf{x}_i$ into the RFF space $\mathbb{R}^D$ is given by

$$\mathbf{z}_i = (D/2)^{-\frac{1}{2}} \left[\cos(\mathbf{v}_1^{\mathrm{T}}\mathbf{x}_i + b_1) \ \ldots \ \cos(\mathbf{v}_D^{\mathrm{T}}\mathbf{x}_i + b_D)\right]^{\mathrm{T}}, \tag{4}$$

where the phase terms $\{b_i\}_{i=1}^D$ are drawn from a uniform distribution on the interval $[0,\ 2\pi]$, and vectors $\{\mathbf{v}_i\}_{i=1}^D$ are drawn from the probability density function (pdf) $p(\mathbf{v})$, which corresponds to the Fourier transform of $k(\mathbf{x}_i - \mathbf{x}_j)$ [14, 19]. To derive the KRG model in the RFF-space, consider the kth entry of the estimate $\mathbf{y}$ as $y_k = \mathbf{w}_k^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x})$ where $\mathbf{w}_k$ denotes the kth column of the parameter matrix $\mathbf{W}$. Using the substitution $\mathbf{W} = \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Psi}$, and the kernel trick $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \boldsymbol{\phi}(\mathbf{x}_i)^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_j)$, we can write

$$y_k = \left(\sum_{n=1}^{N} \Psi_{n,k}\boldsymbol{\phi}(\mathbf{x}_n)\right)^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) = \left(\sum_{n=1}^{N} \Psi_{n,k}\kappa(\mathbf{x}_n, \mathbf{x})\right). \tag{5}$$

Using RFF, (5) can be approximated by

$$y_k \approx \sum_{n=1}^{N} \Psi_{n,k}\mathbf{z}_n^{\mathrm{T}}\mathbf{z} = \mathbf{h}_k^{\mathrm{T}}\mathbf{z}. \tag{6}$$

Finally, the RFF-based regression for the entire graph signal is written as

$$\mathbf{y} = \mathbf{H}^{\mathrm{T}}\mathbf{z}, \tag{7}$$

where $\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2\ \ldots\ \mathbf{h}_K] \in \mathbb{R}^{D\times K}$ is the representation of the regression coefficient matrix in the RFF space. Now, the cost function (3) can be rewritten for the optimization in terms of $\mathbf{H}$ as

$$C(\mathbf{H}) = \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{H}) + \beta \sum_{n=1}^{N} \nu(\mathbf{y}_n). \tag{8}$$

Letting the matrix $\mathbf{Z} = [\mathbf{z}_1\ \mathbf{z}_2\ \ldots\ \mathbf{z}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times D}$ represent the RFF mapping of all training input vectors $\{\mathbf{x}_n\}_{n=1}^N$, the cost function (8) can be rewritten as

$$\begin{aligned} C(\mathbf{H}) = \sum_{n=1}^{N} \|\mathbf{t}_n\|_2^2 - 2\mathrm{tr}(\mathbf{T}^{\mathrm{T}}\mathbf{Z}\mathbf{H}) + \mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}) \\ + \alpha(\mathbf{H}^{\mathrm{T}}\mathbf{H}) + \beta\mathrm{tr}(\mathbf{H}^{\mathrm{T}}\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L}), \end{aligned} \tag{9}$$

where $\mathbf{T} = [\mathbf{t}_1\ \mathbf{t}_2\ \ldots\ \mathbf{t}_N]^{\mathrm{T}} \in \mathbb{R}^{N\times K}$. The gradient of $C(\mathbf{H})$ with respect to $\mathbf{H}$ is given by

$$\nabla C(\mathbf{H}) = -2\mathbf{Z}^{\mathrm{T}}\mathbf{T} + 2\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H} + 2\alpha\mathbf{H} + 2\beta\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L}. \tag{10}$$

By making $\nabla C(\mathbf{H}) = \mathbf{0}$, we obtain

$$(\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \alpha\mathbf{I}_D)\mathbf{H} + \beta\mathbf{Z}^{\mathrm{T}}\mathbf{Z}\mathbf{H}\mathbf{L} = \mathbf{Z}^{\mathrm{T}}\mathbf{T}. \tag{11}$$

Then, vectorizing both sides of (11) and using the relation $\mathrm{vec}(\mathbf{AXB}) = (\mathbf{B}^{\mathrm{T}} \otimes \mathbf{A})\mathrm{vec}(\mathbf{X})$, where $\mathrm{vec}(\cdot)$ denotes the column-stacking operator and $\otimes$ denotes the Kronecker-product operator, the optimum regression coefficients in the RFF space can be obtained as

$$\mathrm{vec}(\mathbf{H}_{\mathrm{o}}) = (\mathbf{B}_{\mathrm{RFF}} + \mathbf{C}_{\mathrm{RFF}})^{-1}\mathrm{vec}(\mathbf{Z}^{\mathrm{T}}\mathbf{T}), \tag{12}$$

where $\mathbf{B}_{\mathrm{RFF}} = (\mathbf{I}_K \otimes (\mathbf{Z}^{\mathrm{T}}\mathbf{Z} + \alpha\mathbf{I}_D))$ and $\mathbf{C}_{\mathrm{RFF}} = (\beta\mathbf{L} \otimes \mathbf{Z}^{\mathrm{T}}\mathbf{Z})$.

Once the regression coefficients are trained, the target estimate $\mathbf{y}$ given an input signal $\mathbf{x}$ corresponding to $\mathbf{z}$ in the RFF space is given by

$$\mathbf{y} = \mathbf{H}_{\mathrm{o}}^{\mathrm{T}}\mathbf{z}.$$

It can be seen that the regression does not depend on the number of training samples and the model has a fixed size D. Therefore, the proposed RFF-approach offers an efficient batch-based KRG for large datasets. Note that the batch-based implementation requires that all samples are available.

4. ONLINE KERNEL REGRESSION ON GRAPHS

We now derive online implementations for KRG using stochastic-gradient descent approaches. These implementations avoid the delay inherent to the batch-based implementation by updating the regression parameters for each new sample. Additionally, each update has a small computational cost when compared to the batch computation.

4.1. Stochastic Gradient Descent KRG

We propose to update the parameters of the KRG in an iterative manner using a stochastic approximation of the gradient $\nabla C(\mathbf{H})$, and derive the stochastic-gradient KRG (SGKRG) algorithm. Consider the instantaneous version, at time n, of the RFF-based KRG model (7), given by

$$\mathbf{y}_n = \mathbf{H}_n^{\mathrm{T}} \mathbf{z}_n, \tag{13}$$

Then, we can write the corresponding instantaneous cost function

$$C(\mathbf{H}_n) = \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \mathrm{tr}(\mathbf{H}_n^{\mathrm{T}} \mathbf{H}_n) + \beta \nu(\mathbf{y}_n). \tag{14}$$

The gradient of $C(\mathbf{H}_n)$ with respect to $\mathbf{H}_n$, which corresponds to a stochastic approximation of $\nabla C(\mathbf{H})$ in (10), is given by

$$\begin{aligned} \nabla C(\mathbf{H}_n) &= -2\mathbf{z}_n \mathbf{t}_n^{\mathrm{T}} + 2\mathbf{z}_n \mathbf{z}_n^{\mathrm{T}} \mathbf{H}_n + 2\alpha \mathbf{H}_n + 2\beta \mathbf{z}_n \mathbf{z}_n^{\mathrm{T}} \mathbf{H}_n \mathbf{L} \\ &= -2\left(\mathbf{z}_n (\mathbf{e}_n^{\mathrm{T}} - \beta \mathbf{y}_n^{\mathrm{T}} \mathbf{L}) - \alpha \mathbf{H}_n\right), \end{aligned} \tag{15}$$

where $\mathbf{e}_n = \mathbf{t}_n - \mathbf{y}_n$ is the network-level error at iteration n. Given (15), in order to recursively minimize $C(\mathbf{H}_n)$, the parameter matrix $\mathbf{H}_n$ is updated by taking a step in the negative-gradient direction as

$$\mathbf{H}_{n+1} = \mathbf{H}_n + \mu\left(\mathbf{z}_n (\mathbf{e}_n^{\mathrm{T}} - \beta \mathbf{y}_n^{\mathrm{T}} \mathbf{L}) - \alpha \mathbf{H}_n\right), \tag{16}$$

where $\mu > 0$ is the step size. Equation (16) represents the update equation for the proposed online SGKRG algorithm using RFF. We note the gradient obtained using all samples is expected to offer a better optimization direction than the one obtained using a single sample. Thus, the SGKRG and the batch-based KRG have opposite characteristics in the trade-off between complexity and convergence speed.

5. CONVERGENCE ANALYSIS

In this section, we analyze the convergence of the proposed online algorithm in the mean sense [20–22]. We assume the observation model $\mathbf{t}_n = \mathbf{H}_{\mathrm{o}}^{\mathrm{T}} \mathbf{z}_n + \boldsymbol{\upsilon}_n$, where $\boldsymbol{\upsilon}_n$ is observation noise vector. The noise is assumed to be zero-mean and independent of $\mathbf{z}_n$. Let $\boldsymbol{\Xi}_n = \mathbf{H}_n - \mathbf{H}_{\mathrm{o}}$ denote the deviation between the regression parameters $\mathbf{H}_n$ and the optimal parameters $\mathbf{H}_{\mathrm{o}}$ at iteration n. We have

$$\begin{aligned} \boldsymbol{\Xi}_{n+1} &= \mathbf{H}_n + \mu\left(\mathbf{z}_n (\mathbf{t}_n^{\mathrm{T}} - \mathbf{z}_n^{\mathrm{T}} \mathbf{H}_n - \beta \mathbf{z}_n^{\mathrm{T}} \mathbf{H}_n \mathbf{L}) - \alpha \mathbf{H}_n\right) - \mathbf{H}_{\mathrm{o}} \\ &= \boldsymbol{\Xi}_n + \mu\left(\mathbf{z}_n (\mathbf{t}_n^{\mathrm{T}} - \mathbf{z}_n^{\mathrm{T}} \mathbf{H}_n) - \beta \mathbf{z}_n \mathbf{z}_n^{\mathrm{T}} \mathbf{H}_n \mathbf{L} - \alpha \mathbf{H}_n\right), \end{aligned} \tag{17}$$

which can be rewritten as

$$\begin{aligned} &\boldsymbol{\Xi}_{n+1} \\ &= \boldsymbol{\Xi}_n + \mu(\mathbf{z}_n \boldsymbol{\upsilon}_n^{\mathrm{T}} + \mathbf{z}_n \mathbf{z}_n^{\mathrm{T}} (\mathbf{H}_{\mathrm{o}} - \mathbf{H}_n) - \beta \mathbf{z}_n \mathbf{z}_n^{\mathrm{T}} \mathbf{H}_n \mathbf{L} - \alpha \mathbf{H}_n) \\ &= (\mathbf{I}_D - \mu \mathbf{z}_n \mathbf{z}_n^{\mathrm{T}}) \boldsymbol{\Xi}_n + \mu \mathbf{z}_n \boldsymbol{\upsilon}_n^{\mathrm{T}} - \mu(\beta \mathbf{z}_n \mathbf{z}_n^{\mathrm{T}} \mathbf{H}_n \mathbf{L} + \alpha \mathbf{H}_n). \end{aligned} \tag{18}$$

Vectorizing both sides of (18), we obtain

$$\begin{aligned} \boldsymbol{\xi}_{n+1} = &\left(\mathbf{I}_K \otimes (\mathbf{I}_D - \mu \mathbf{z}_n \mathbf{z}_n^{\mathrm{T}})\right) \boldsymbol{\xi}_n + \mu(\mathbf{I}_K \otimes \mathbf{z}_n) \boldsymbol{\upsilon}_n \\ &- \mu\left(\alpha \mathbf{I}_{KD} + (\beta \mathbf{L} \otimes \mathbf{z}_n \mathbf{z}_n^{\mathrm{T}})\right) \mathrm{vec}(\mathbf{H}_n), \end{aligned} \tag{19}$$

where $\boldsymbol{\xi}_n = \mathrm{vec}(\boldsymbol{\Xi}_n)$. Substituting $\mathrm{vec}(\mathbf{H}_n) = \boldsymbol{\xi}_n + \mathrm{vec}(\mathbf{H}_{\mathrm{o}})$ into (19), it can be rewritten as

$$\begin{aligned} \boldsymbol{\xi}_{n+1} = &\left[\mathbf{I}_{KD} - \mu\left(\alpha \mathbf{I}_{KD} + (\mathbf{I}_K + \beta \mathbf{L}) \otimes (\mathbf{z}_n \mathbf{z}_n^{\mathrm{T}})\right)\right] \boldsymbol{\xi}_n \\ &- \mu\left(\alpha \mathbf{I}_{KD} + \beta \mathbf{L} \otimes (\mathbf{z}_n \mathbf{z}_n^{\mathrm{T}})\right) \mathrm{vec}(\mathbf{H}_{\mathrm{o}}) \\ &+ \mu(\mathbf{I}_K \otimes \mathbf{z}_n) \boldsymbol{\upsilon}_n. \end{aligned} \tag{20}$$

We now take the expected value on both sides of (20). Given the zero-mean and independence assumptions on the observation noise, the last term's expected value on the right-hand side of (20) is zero. We obtain the following recursion on $\mathrm{E}[\boldsymbol{\xi}_n]$

$$\mathrm{E}[\boldsymbol{\xi}_{n+1}] = \boldsymbol{\mathcal{A}} \mathrm{E}[\boldsymbol{\xi}_n] - \boldsymbol{\mathcal{B}} \mathrm{vec}(\mathbf{H}_{\mathrm{o}}), \tag{21}$$

where

$$\begin{aligned} \boldsymbol{\mathcal{A}} &= \mathbf{I}_{KD} - \mu\left(\alpha \mathbf{I}_{KD} + (\mathbf{I}_K + \beta \mathbf{L}) \otimes \mathbf{R}_{\mathbf{z}}\right) \\ \boldsymbol{\mathcal{B}} &= \mu\left(\alpha \mathbf{I}_{KD} + \beta \mathbf{L} \otimes \mathbf{R}_{\mathbf{z}}\right), \end{aligned} \tag{22}$$

with $\mathbf{R}_{\mathbf{z}} = \mathrm{E}[\mathbf{z}_n \mathbf{z}_n^{\mathrm{T}}]$. Taking the recursion (21) down to zero, we obtain

$$\mathrm{E}[\boldsymbol{\xi}_n] = \boldsymbol{\mathcal{A}}^n \mathrm{E}[\boldsymbol{\xi}_0] - \mu \sum_{i=0}^{n-1} \boldsymbol{\mathcal{A}}^{n-1-i} \boldsymbol{\mathcal{B}} \mathrm{vec}(\mathbf{H}_{\mathrm{o}}). \tag{23}$$

From (23), we see that convergence is guaranteed if $\lim_{n\to\infty} \boldsymbol{\mathcal{A}}^n = 0$, which is achieved when $\rho(\boldsymbol{\mathcal{A}}) < 1$, where $\rho(\cdot)$ denotes the spectral radius of the argument, i.e., its largest absolute eigenvalue. We have that $\rho(\boldsymbol{\mathcal{A}}) < 1$ if $\rho\left(\mu\left(\alpha \mathbf{I}_{KD} + (\mathbf{I}_K + \beta \mathbf{L}) \otimes \mathbf{R}_{\mathbf{z}}\right)\right) < 2$. Therefore, a sufficient condition for the convergence of the proposed SGKRG algorithms is given by

$$0 < \mu < \frac{2}{\rho(\mathbf{R}_{\mathbf{z}}) + \alpha + \beta \rho(\mathbf{L}) \rho(\mathbf{R}_{\mathbf{z}})}. \tag{24}$$

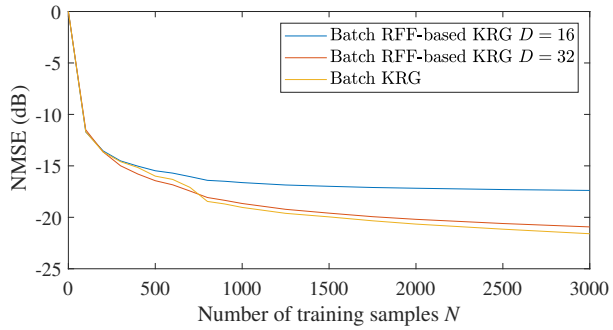
Under the convergence condition (24), (23) converges asymptotically to $(\mathbf{I}_{KD} - \boldsymbol{\mathcal{A}})^{-1} \boldsymbol{\mathcal{B}} \mathrm{vec}(\mathbf{H}_{\mathrm{o}})$, which reduces to $(\alpha \mathbf{I}_{KD} + (\mathbf{I}_K + \beta \mathbf{L}) \otimes \mathbf{R}_z)^{-1} (\alpha \mathbf{I}_{KD} + \beta \mathbf{L} \otimes \mathbf{R}_z) \mathrm{vec}(\mathbf{H}_{\mathrm{o}})$. This means that the solution of the SGKRG is asymptotically biased in the mean sense. The bias is introduced by the regularization coefficients α and β.

6. NUMERICAL EXPERIMENTS

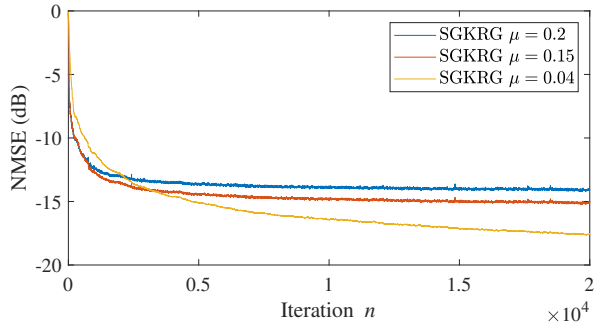
In this section we validate the performance of the proposed algorithms. We reproduce two regression problems adopted in [16] and compare the proposed algorithms against the conventional KRG. In both experiments, we use the Gaussian kernel $\kappa(\mathbf{x}_i, \mathbf{x}_j) = \exp\left(-\|\mathbf{x}_i - \mathbf{x}_j\|_2^2 / (2\sigma^2)\right)$ when employing the kernel methods and the RFF implementations.

6.1. Synthesized Data

The first experiment uses an Erdös Rényi graph with artificially generated data. The graph has $K = 50$ nodes with edge-probability equal to 0.1. A total of $S = 20000$ K-dimensional data samples are generated as follows. First, a covariance matrix $\mathbf{C}_S \in \mathbb{R}^{S \times S}$ is drawn from the inverse Wishart distribution with an identity hyperparameter matrix and S degrees of freedom. Then, K independent vector realizations are drawn from an S-dimensional Gaussian distribution $\mathcal{N}(\mathbf{0}, \mathbf{C}_S)$. Letting $\{\mathbf{x}_n\}_{n=1}^S$ denote the obtained signals, the corresponding target vectors $\{\mathbf{t}_n\}_{n=1}^S$ are generated by projecting each signal onto the graph by solving $\mathbf{t}_n = \arg\min_{\boldsymbol{\tau}} \left\{\|\mathbf{x}_n - \boldsymbol{\tau}\|_2^2 + \boldsymbol{\tau}^{\mathrm{T}} \mathbf{L} \boldsymbol{\tau}\right\}$.



(a) Batch-based solutions.



(b) Online solution.

Fig. 1. NMSE achieved by the KRG implementations versus number of training samples.

From the total $S = 20000$ samples, we use 1000 samples as test dataset and up to 19000 samples as training dataset. Target data in the training dataset are corrupted with additive white Gaussian noise (AWGN) and the signal-to-noise ratio is 5 dB. The parameters α and β are estimated via 5-fold cross-validation with grid-search using a separate dataset, and minimizing the normalized squared estimation error

$$\text{NMSE} = 10 \log_{10} \left(\text{E} \left[\frac{\|\mathbf{Y} - \mathbf{T}_0\|_{\text{F}}^2}{\|\mathbf{T}_0\|_{\text{F}}^2} \right] \right), \qquad (25)$$

where $\mathbf{T}$ is the true target matrix, $\mathbf{Y}$ is the estimated matrix. We select the parameters that result in the best NMSE at the end of the learning process.

The proposed batch-based algorithm is compared against the conventional KRG from [16] and results, averaged over 500 independent runs, are presented in Fig. 1a. The batch-based algorithms are trained with up to 3000 samples, which meets the limit of our computational power when running the conventional KRG. Plots show that the RFF-based approach approximates well the conventional KRG. In Fig. 1b, results show that the online algorithm is capable of learning the regression parameters. We run the SGKRG up to 20000 samples, such that the test samples are included at the end of the initial training samples. Note that the value of μ affects the convergence of the proposed stochastic-gradient-based implementation. We demonstrate the performance for different step sizes and we show that the NMSE level achieved by the SGKRG approximates that of the batch RFF-based KRG as μ decreases.

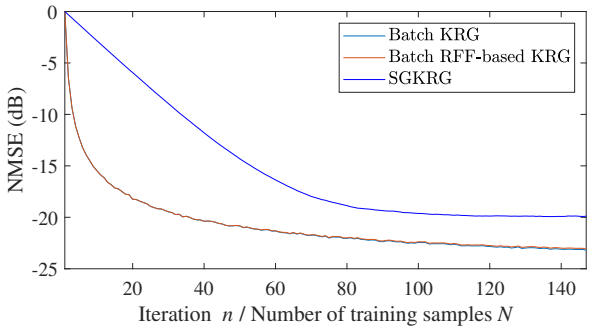


Fig. 2. NMSE achieved by the KRG implementations versus number of training samples for the fMRI signal simulation.

6.2. Real Data

The second experiment uses real data and addresses the task of estimating brain activity of voxels in a functional magnetic resonance imaging (fMRI) dataset. The data and graph used in [16] are made available in [23] and the same are used in this experiment.

A voxel is a volumetric unit that constitutes a 3-dimensional image of the brain, and each voxel is associated with a small cubic portion of the brain. Regions of the brain relate to each other anatomically and, by considering these relations, a graph can be constructed where voxels are the nodes, and edges represent relations between them. More details on this dataset and graph construction are provided in [16]. The regression experiment consists of estimating the signal on 90 of the voxels using the signal from 10 other voxels, such that the graph structure corresponds to the set of pairwise relations of the 90 voxels. Training and test datasets have the same size equal to 146 input-target pairs. The training signal is corrupted by an AWGN with covariance matrix $0.1 \cdot \mathbf{I}_K$.

We conduct 100 independent runs with different permutations of signals between training and test datasets. The RFF-space dimension is $D = 32$. Results are shown in Fig. 2 for both batch and online algorithms. In this experiment, both batch-based implementations converge together to approximately -23 dB and it can be observed that the RFF-based implementation matches the conventional KRG. Results show that the SG-based implementation is capable of successfully learning the target model and achieving low NMSE, around -20 dB, using $\mu = 0.04$.

7. CONCLUSION

This paper proposed batch-based and online implementations for kernel regression on graphs. The performance of the proposed algorithms was validated with numerical experiments using synthesized and real data. The proposed batch-based implementation uses RFF to approximate kernel evaluations as inner-products in a fixed-dimension space, reducing the complexity of the regression model compared to the conventional KRG. Results of numerical experiments showed no significant performance loss in approximating the kernel evaluations using RFF, such that the RFF-based implementation can achieve the same performance as the conventional KRG. Additionally, sufficient conditions for convergence of this algorithm in the mean sense were derived.

8. REFERENCES

[1] A. Sandryhaila and J. M. F. 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, pp. 80–90, Sep. 2014.

[2] 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, pp. 83–98, May 2013.

[3] B. Boucher and S. Jenna, "Genetic interaction networks: better understand to better predict," *Front. in Genet.*, vol. 4, pp. 1–16, Dec. 2013.

[4] W. Huang, T. A. W. Bolton, J. D. Medaglia, D. S. Bassett, A. Ribeiro, and D. Van De Ville, "A graph signal processing perspective on functional brain imaging," *Proc. IEEE*, vol. 106, pp. 868–885, May 2018.

[5] I. Jablonski, "Graph signal processing in applications to sensor networks, smart grids, and smart cities," *IEEE Sensors J.*, vol. 17, pp. 7659–7666, Dec. 2017.

[6] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs," *IEEE Trans. Signal Process.*, vol. 61, pp. 1644–1656, Apr. 2013.

[7] A. Ortega, P. Frossard, J. Kovacevic, J. M. F. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proc. IEEE*, vol. 106, pp. 808–828, May 2018.

[8] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Graph filters," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, May 2013, pp. 6163–6166.

[9] A. Gavili and X.-P. Zhang, "On the shift operator, graph frequency, and optimal filtering in graph signal processing," *IEEE Trans. Signal Process.*, vol. 65, pp. 6303–6318, Dec. 2017.

[10] 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, June 2018.

[11] D. Thanou, D. I. Shuman and P. Frossard, "Learning Parametric Dictionaries for Signals on Graphs," *IEEE Trans. Signal Process.*, vol. 62, no. 15, pp. 3849-3862, Aug 2014.

[12] R. Nassif, C. Richard, J. Chen, and A. H. Sayed, "Distributed diffusion adaptation over graph signals," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, 2018, pp. 4129–4133.

[13] F. Hua, R. Nassif, C. Richard, H. Wang and A. H. Sayed, "Online distributed learning over graphs with multitask graph-filter models," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 6, pp. 63–77, Jan. 2020.

[14] 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.

[15] D. Romero, V. N. Ioannidis and G. B. Giannakis, "Kernel-based reconstruction of space-time functions on dynamic graphs," *IEEE J. Sel. Top. Signal Process.*, vol. 11, no. 6, pp. 856–869, Sep. 2017.

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

[17] Y. Shen, G. Leus, and G. B. Giannakis, "Online graph-adaptive learning with scalability and privacy," *IEEE Trans. Signal Process.*, vol. 67, pp. 2471–2483, May 2019.

[18] V. G. Gogineni, V. R. M. Elias, W. A. Martins and S. Werner, "Graph diffusion kernel LMS using random Fourier Features," in *Conf. Rec. Asilomar Conf. Signals Syst. Comput.*, pp. 1–5, Nov. 2020.

[19] A. Rahimi and B. Recht, "Random features for large-scale kernel machines," in *Proc. Adv. Neural Inf. Process. Syst.*, pp. 1177–1184, 2007.

[20] W. A. Gardner, "Learning characteristics of stochastic-gradient-descent algorithms: A general study, analysis, and critique," *Signal Process.*, vol 6, no. 2, pp. 113–133, 1984.

[21] A. H. Sayed, *Adaptive Filters*. Wiley, Jan. 2008.

[22] P. Di Lorenzo, S. Barbarossa, P. Banelli and S. Sardellitti, "Adaptive Least Mean Squares Estimation of Graph Signals," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 2, no. 4, pp. 555–568, Dec. 2016.

[23] "Reproducible Research | KTH", `https://www.kth.se/ise/research/reproducibleresearch-1.433797`, 2020. Accessed: 2020-08-18.