KERNEL REGRESSION FOR GRAPH SIGNAL PREDICTION IN PRESENCE OF SPARSE NOISE

Arun Venkitaraman[*]*, Pascal Frossard*[†]*, and Saikat Chatterjee*[*]

[*] Department of Information Science and Engineering,
School of Electrical Engineering and Computer Science, KTH Royal Institute of Technology, Sweden
[†] Signal Processing Laboratory (LTS4), Ecole Polytechnique Fédérale de Lausanne (EPFL)
arunv@kth.se, pascal.frossard@epfl.ch, sach@kth.se

ABSTRACT

In presence of sparse noise we propose kernel regression for predicting output vectors which are smooth over a given graph. Sparse noise models the training outputs being corrupted either with missing samples or large perturbations. The presence of sparse noise is handled using appropriate use of ℓ_1-norm along-with use of ℓ_2-norm in a convex cost function. For optimization of the cost function, we propose an iteratively reweighted least-squares (IRLS) approach that is suitable for kernel substitution or kernel trick due to availability of a closed form solution. Simulations using real-world temperature data show efficacy of our proposed method, mainly for limited-size training datasets.

***Index Terms*—** Kernel regression, graph signal processing, Sparse noise, graph-Laplacian, iteratively reweighted least squares

1. INTRODUCTION

Kernel regression deals with the problem of learning a model relating a given input vector to its associated output vector, such that the learnt model provides a prediction for the output, when given an input. The model is learnt using a training dataset of input-output pairs and is based on a kernel function associating the different inputs. The kernel function captures the similarity between the various input points over a high-dimensional space. In the case when the kernel function represents is given by the inner-product between input vectors, we get the more popular special case of linear regression [1]. Kernel regression plays a central role in a gamut of techniques from support vector machines [1], extreme learning machines [2] to deep learning [3]. More recently, kernel-based approaches have been applied to the analysis of graph signals.

One of the earliest works which employ graph-regularization in conjunction with kernels is due to Smola et al. [4], though it was not in the context of graph signal processing. Kernel-based reconstruction of graph signals was proposed by Romero et al. in [5, 6] where both the input and output are graph signal values at subsets of nodes of a given graph. This was then further extended to joint space-time graph signal reconstruction in [7–9]. Using the framework of diffusion wavelets, Chung et al employed kernel regression in mandible growth modeling in CT images [10]. Shen et al. proposed the use of kernels in structural equation models for identifying network topologies from graph signals [11]. Kernel regression for graph signal outputs with inputs agnostic to a graph was considered first by Venkitaraman et al. [12], which they later extended to its Bayesian version in the form of Gaussian processes over graphs [13]. The related problem of using multi-kernel approaches which allow selection of kernels directly from data have also been pursued in the context of graph signals [14, 15].

However, most of the aforementioned works are formulated for either the case when there is no noise or that the noise is implicitly Gaussian distributed, which may not be always a realistic assumption. In many scenarios in practice, noise occurs usually at a sparse subset of nodes, either due to samples being missing or due to random large perturbations in the signal value. In such cases, the performance of these approaches becomes severely limited. Motivated by this observation, we propose kernel regression for predicting signals which are smooth over an associated graph, when the training data is scarce and noisy with the training outputs corrupted with sparse noise. By sparse noise, we mean the case where each training output may have an unknown subset of nodes with either missing samples or large perturbations in the signal value. In such adverse scenarios, applying kernel regression on each node individually would not yield a reasonable prediction performance. The use of structural information significantly boosts prediction performance in such case. Since we consider outputs which are graph signals, we use the graph smoothness to improve prediction.

We first start from the first principles of linear regression and learn the optimal regression coefficients by minimizing a cost consisting of both graph smoothness constraint and the ℓ_1 norm of the model error, since we assume sparse noise. A direct minimization of the cost does not admit a closed-form solution necessary to arrive at the more general kernel based formulation. The kernel formulation is desirable over linear regression since it is usually unclear as to what is the best feature to be applied on the input used in linear regression. On the other hand, it is usually easier and more intuitive to express the relation (similarity/ dissimilarity) between various input observations using a kernel function. In order that the proposed linear regression further results in a kernel regression formulation, we take an iteratively reweighted least squares (IRLS) approach in solving the ℓ_1-norm based optimization.

2. KERNEL REGRESSION OVER GRAPHS IN PRESENCE OF SPARSE NOISE

We first introduce the relevant background in graph signal processing and proceed to our problem formulation.

2.1. Brief review of graph signal processing basics

Consider a graph G with M nodes, with the edge set $\mathcal{E}$ and the adjacency matrix $\mathbf{A}$, where $\mathbf{A}(i,j)$ denotes the strength of the edge between the ith and jth nodes. We consider undirected

graphs with symmetric edge-weights or $\mathbf{A} = \mathbf{A}^\top$. The graph-Laplacian matrix $\mathbf{L}$ of G is given by $\mathbf{L} = \mathbf{D} - \mathbf{A}$, where $\mathbf{D}$ is the diagonal degree matrix with ith diagonal element given by the sum of the elements in the ith row of $\mathbf{A}$. A graph signal $\mathbf{y} = [y(1)\, y(2)\, \cdots y(M)]^\top \in \mathbb{R}^M$ is a vector whose components are the values of a physical quantity at the various nodes of $\mathcal{G}$. The quadratic form $\mathbf{y}^\top \mathbf{L}\mathbf{y} = \sum_{(i,j)\in\mathcal{E}} \mathbf{A}(i,j)(y(i) - y(j))^2$ is usually used to quantify the smoothness of $\mathbf{y}$ over the graph: a small value indicates that $\mathbf{y}$ has similar values across connected nodes. Such a signal $\mathbf{y}$ is referred to as a smooth graph signal. Graph signal processing is a continuously expanding field of research, and we refer the reader to [16, 17] and the references therein for a more comprehensive study of the framework.

2.2. Kernel Regression over Graphs in Presence of Sparse Noise

Let $\{\mathbf{x}_n \in \mathbb{R}^{N_i}, \mathbf{t}_n \in \mathbb{R}^M\}_{n=1}^N$ denote the training dataset comprising N input-output pairs, N_i and M being the dimensions of the input and output, respectively. We consider outputs $\mathbf{t}_n$ which are smooth over a known graph of M nodes with the graph-Laplacian matrix $\mathbf{L}$. We assume that the training outputs are corrupted with sparse noise – either through missing signal values or large perturbations in signal values at an unknown subset of nodes, when nothing is known about the nodes at which this corruption occurs. The corrupted nodes are also not assumed to be the same across different training observations. Our primary goal is to model the graph signal output $\mathbf{t}_n$ with a kernel regression model, so as to make predictions for the output when a new test input is given to the model. In order to achieve this, we start from the case of linear regression and proceed to derive the kernel regression case.

2.3. Linear basis model for regression over graphs

The predicted output $\mathbf{y}_n$ of linear regression for $\mathbf{x}_n$ is given by

$$\mathbf{y}_n = \mathbf{W}^\top \boldsymbol{\phi}(\mathbf{x}_n), \tag{1}$$

where $\boldsymbol{\phi}(\mathbf{x}_n) \in \mathbb{R}^K$ is some pre-defined function of $\mathbf{x}_n$ and $\mathbf{W} \in \mathbb{R}^{K\times M}$ denotes the regression coefficient matrix. Equation (1) is referred to as linear basis model for regression, usually termed linear regression for brevity (cf. Chapters 3 and 6 in [1]). Given the output model, we learn the optimal parameter matrix $\mathbf{W}$ by minimizing the following cost function:

$$C_1(\mathbf{W}) = \sum_{n=1}^N \|\mathbf{t}_n - \mathbf{y}_n\|_1 + \alpha \mathrm{tr}(\mathbf{W}^\top\mathbf{W}) + \beta \sum_{n=1}^N \mathbf{y}_n^\top \mathbf{L}\mathbf{y}_n, \tag{2}$$

where regularization coefficients $\alpha, \beta \geq 0$, $\mathrm{tr}(\cdot)$ denotes the trace operator, and $\|\mathbf{x}\|_p$ the ℓ_p norm of $\mathbf{x}$. Since we assume that the corruptions occur due to presence of a sparse noise, we minimize the ℓ_1 norm of the model error. We use the ℓ_1 norm as it is a well-known convex relaxation of the ℓ_0 norm and it aids analytical tractability.

The graph-Laplacian based regularization promotes the predicted output to be smooth over the given graph. Finally, the regularization $\mathrm{tr}(\mathbf{W}^\top\mathbf{W}) = \|\mathbf{W}\|_F^2$ ensures that $\mathbf{W}$ remains bounded. We refer to (2) as the *linear regression over graphs for sparse noise* (LRGS). Although LRGS is a convex cost, it does not result in a closed-form solution. As a result, a direct kernel regression model is not possible for LRGS, since the kernel substitution or kernel trick cannot be employed in the absence of a closed form solution. In this paper, we adopt the iteratively reweighted least squares approach (IRLS) to solve ℓ_1 minimization problems described in the next Section. This is because IRLS helps translate (2) into a more general kernel regression approach.

2.4. Iteratively reweighted least squares (IRLS)

Iteratively reweighted least squares (IRLS) is a popular approach for solving ℓ_1 minimization problems iteratively [18, 19]. IRLS solves an ℓ_1 minimization problem by successively solving a series of weighted least-squares problems, with weights that depend on values from the previous iteration. Consider the general problem for $\mathbf{z}^* = \arg \mathbf{z} \in \mathbf{R}^N$:

$$\min \|\mathbf{z}\|_1,\ \mathbf{z} \in \mathcal{S}$$

where $\mathcal{S} \subset \mathbf{R}^N$ is the subspace in which $\mathbf{z}$ lies. Then, IRLS iteratively solves for optimal $\mathbf{z}$ as follows:

$$\begin{aligned} &\mathbf{z}_n^* = \arg\min \|\mathbf{D}_n\mathbf{z}_n\|_2^2,\ \mathbf{z} \in \mathcal{S} \\ &\text{s.t. } \mathbf{D}_n = \mathrm{diag}(d_{n,1}, \cdots, d_{n,N}), \end{aligned}$$

where $d_{n,i} = (z_{n-1}(i) + \delta)^{-1/2}$ and $\delta > 0$ is a small constant introduced to circumvent ill-conditioning. In the case of convergence $\mathbf{z}_{n-1} \approx \mathbf{z}_n$ and then $\|\mathbf{D}_n\mathbf{z}_n\|_2^2 = \sum_{i=1}^N z_n^2(i) z_{n-1}^{-1}(i) \approx \sum_{i=1}^N |z_n(i)| = \|\mathbf{z}\|_1$. Thus, IRLS assymptotically solves an ℓ_1 minimization problem.

2.5. Solving LRGS using IRLS

We now propose to solve (2) using IRLS by solving the following cost iteratively:

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

where $\mathbf{D}_n$ is the diagonal IRLS weighting matrix for the nth training observation such that

$$d_n(i) = |(t_{n-1}(i) - y_{n-1}(i)) + \delta|^{-1/2}. \tag{4}$$

We define matrices $\mathbf{T}$, $\mathbf{Y}$ and $\boldsymbol{\Phi}$ as follows:

$$\begin{aligned} \mathbf{Y} &= [\mathbf{y}_1\, \mathbf{y}_2\, \cdots \mathbf{y}_N]^\top \in \mathbb{R}^{N\times M}, \\ \boldsymbol{\Phi} &= [\boldsymbol{\phi}(\mathbf{x}_1)\, \boldsymbol{\phi}(\mathbf{x}_2)\, \cdots \boldsymbol{\phi}(\mathbf{x}_N)]^\top \in \mathbb{R}^{N\times K}, \\ \mathbf{T}_D &= [\mathbf{t}_{1,D}\, \mathbf{t}_{2,D}\, \cdots \mathbf{t}_{N,D}]^\top \in \mathbb{R}^{N\times M}, \end{aligned} \tag{5}$$

where $\mathbf{t}_{n,D} = \mathbf{D}_n^2 \mathbf{t}_n$. Using (1) and (5), the cost function (3) is expressible as

$$\begin{aligned} C(\mathbf{W}) \quad &= \sum_n \|\mathbf{D}_n\mathbf{t}_n\|_2^2 - 2\,\mathrm{tr}\left(\mathbf{T}_D^\top \boldsymbol{\Phi}\mathbf{W}\right) \\ &+ \mathrm{tr}\left(\textstyle\sum_n \mathbf{W}^\top \boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^\top \mathbf{W}\mathbf{D}_n^2\right) \\ &+ \alpha\,\mathrm{tr}(\mathbf{W}^\top\mathbf{W}) + \beta\,\mathrm{tr}\left(\mathbf{W}^\top\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\mathbf{W}\mathbf{L}\right). \end{aligned} \tag{6}$$

where we use properties of the matrix trace operation. Since the cost function is quadratic in $\mathbf{W}$, we get the globally optimal and unique solution by setting the gradient of C with respect to $\mathbf{W}$ equal to zero. Setting $\dfrac{\partial C}{\partial \mathbf{W}} = 0$, we get that

$$\begin{aligned} &-\boldsymbol{\Phi}^\top\mathbf{T}_D + \sum_n \boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^\top\mathbf{W}\mathbf{D}_n^2 + \alpha\mathbf{W} \\ &\quad + \beta\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\mathbf{W}\mathbf{L} = \mathbf{0}, \text{or,} \\ &\alpha\mathbf{W} + \beta\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\mathbf{W}\beta\mathbf{L} + \sum_n \boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^\top\mathbf{W}\mathbf{D}_n^2 = \boldsymbol{\Phi}^\top\mathbf{T}_D, \end{aligned} \tag{7}$$

which on vectorizing and rearranging, gives the optimal parameter matrix $\mathbf{W}_\star$ as:

$$\begin{aligned}&\mathrm{vec}(\mathbf{W}_\star)\\&=\left[\alpha\mathbf{I}_{MN}+\sum_n \mathbf{D}_n^2\otimes\boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^\top+\beta\mathbf{L}\otimes\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\right]^{-1}\\&\quad\times\mathrm{vec}(\boldsymbol{\Phi}^\top\mathbf{T}_D),\end{aligned}$$

where $\mathrm{vec}(\cdot)$ denotes the standard vectorization operator and $\otimes$ denotes the Kronecker product operation [20]. The predicted output for a new input $\mathbf{x}$ is then given by

$$\mathbf{y}=\mathbf{W}_\star^\top\boldsymbol{\phi}(\mathbf{x}).$$

Our development of linear regression so far seemingly assumes that we have access to an input feature function $\boldsymbol{\phi}(\cdot)$. In many practical cases, it is usually not clear what such a function should so that it helps achieve a reasonable prediction model through Eq (1). On the other hand, it is generally more clear as to what kind of association is reasonable between various inputs. This can be usually captured in the form of a kernel function between input points, for example with radial-basis functions or Gaussian kernels. This makes kernel regression more intuitive and favourable in designing good predictors than linear regression. We next show that the approach we have pursued so far does not actually require an explicit $\boldsymbol{\phi}(\cdot)$, since it only uses inner-products of the form $\boldsymbol{\phi}(\mathbf{x}_m)^\top\boldsymbol{\phi}(\mathbf{x}_n)$. Hence, the results may be reformulated completely in terms of kernel functions by substituting the inner product with general kernel functions $k(\mathbf{x}_m,\mathbf{x}_n)$– thus leading to the kernel regression over graphs for sparse noise.

2.6. Kernel regression over graphs for sparse noise (KRGS)

We now demonstrate how IRLS approach to LRGS naturally leads to the more general kernel regression problem, that is, to kernel regression over graphs for sparse noise (KRGS). To achieve this, we use the substitution $\mathbf{W}=\boldsymbol{\Phi}^\top\boldsymbol{\Psi}$ and express the cost function in terms of the parameter $\boldsymbol{\Psi}\in\mathbb{R}^{N\times M}$. This substitution is motivated by observing that on rearranging the terms in (8), we get that

$$\mathbf{W}=\boldsymbol{\Phi}^\top\boldsymbol{\Psi},$$

where $\boldsymbol{\Psi}=\left[\frac{1}{\alpha}\left(\mathbf{T}_D-\beta\boldsymbol{\Phi}\mathbf{W}\mathbf{L}-\boldsymbol{\Phi}_D\right)\right]$, such that $\boldsymbol{\Phi}_D\in\mathbb{R}^{N\times M}$ is the matrix whose nth row is given by $\boldsymbol{\phi}(\mathbf{x}_n)^\top\mathbf{W}\mathbf{D}_n^2$. On substituting $\mathbf{W}=\boldsymbol{\Phi}^\top\boldsymbol{\Psi}$ in Equation (6), $\boldsymbol{\Psi}$ becomes the parameter matrix for the dual cost:

$$\begin{aligned}C(\boldsymbol{\Psi})=&-2\mathrm{tr}\left(\mathbf{T}_D^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{\Psi}\right)\\&+\mathrm{tr}\left(\sum_n\boldsymbol{\Psi}^\top\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^\top\boldsymbol{\Phi}^\top\boldsymbol{\Psi}\mathbf{D}_n^2\right)\\&+\alpha\,\mathrm{tr}(\boldsymbol{\Psi}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{\Psi})+\beta\,\mathrm{tr}\left(\boldsymbol{\Psi}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\boldsymbol{\Psi}\mathbf{L}\right).\end{aligned}\tag{8}$$

Kernel substitution or the kernel trick refers to the procedure in regression wherein all inner-products of feature vectors are substituted by a more general kernel function [1]. That is, one replaces all inner-products of the form $\boldsymbol{\phi}(\mathbf{x}_m)^\top\boldsymbol{\phi}(\mathbf{x}_n)$ with a kernel between inputs $\mathbf{x}_m$ and $\mathbf{x}_n$ given by $k_{m,n}=k(\mathbf{x}_m,\mathbf{x}_n)$, where $k(\cdot)$ is some valid kernel function, such as the radial-basis function or the Gaussian kernel. Correspondingly, the matrix $\boldsymbol{\Phi}\boldsymbol{\Phi}^\top\in\mathbb{R}^{N\times N}$ is generalized to the kernel matrix $\mathbf{K}$ between training samples such that its (m,n)th entry is given by $k_{m,n}$. Further let us denote $\mathbf{k}(\mathbf{x})=[k_1(\mathbf{x}),k_2(\mathbf{x}),\cdots,k_N(\mathbf{x})]^\top$ and $k_n(\mathbf{x})=k(\mathbf{x}_m,\mathbf{x})$. Then, we have that

$$\begin{aligned}C(\boldsymbol{\Psi})=&-2\mathrm{tr}\left(\mathbf{T}_D^\top\mathbf{K}\boldsymbol{\Psi}\right)+\mathrm{tr}\left(\sum_n\boldsymbol{\Psi}^\top\mathbf{k}(\mathbf{x}_n)\mathbf{k}(\mathbf{x}_n)^\top\boldsymbol{\Psi}\mathbf{D}_n^2\right)\\&+\alpha\,\mathrm{tr}(\boldsymbol{\Psi}^\top\mathbf{K}\boldsymbol{\Psi})+\beta\,\mathrm{tr}\left(\boldsymbol{\Psi}^\top\mathbf{K}\mathbf{K}\boldsymbol{\Psi}\mathbf{L}\right),\end{aligned}\tag{9}$$

In kernel regression literature, (9) is termed a dual representation of Equation (6) (cf. Chapter 6 of [1]). Thus, we see that the entire regression problem is generalized to the usage of arbitrary kernel functions, without the need of an explicit input feature $\boldsymbol{\phi}$. Setting the derivative of $C(\boldsymbol{\Psi})$ with respect to $\boldsymbol{\Psi}$ to zero, we get that

$$\begin{aligned}\mathbf{0}=\frac{\partial C(\boldsymbol{\Psi})}{\partial\boldsymbol{\Psi}}=&-2\mathbf{K}\mathbf{T}_D\boldsymbol{\Psi}+2\sum_n\mathbf{k}(\mathbf{x}_n)\mathbf{k}(\mathbf{x}_n)^\top\boldsymbol{\Psi}\mathbf{D}_n^2\\&+2\alpha\,\mathbf{K}\boldsymbol{\Psi}+2\beta\mathbf{K}\mathbf{K}\boldsymbol{\Psi}\mathbf{L}.\end{aligned}\tag{10}$$

On vectorizing both sides of (10) and rearranging , we get that

$$\begin{aligned}&\mathrm{vec}(\boldsymbol{\Psi})=\mathbf{B}\mathrm{vec}(\mathbf{T}_D),\ \text{where}\\&\mathbf{B}=\left[\sum_n\mathbf{D}_n^2\otimes\mathbf{k}(\mathbf{x}_n)\mathbf{k}(\mathbf{x}_n)^\top+\alpha(\mathbf{I}_M\otimes\mathbf{K})+(\beta\mathbf{L}\otimes\mathbf{K}^2)\right]^{-1}\\&\quad\times(\mathbf{I}_M\otimes\mathbf{K}).\end{aligned}\tag{11}$$

Using the $\boldsymbol{\Psi}$ so computed, the output of the kernel regression for a new test input $\mathbf{x}$ is given by

$$\begin{aligned}\mathbf{y}&=\mathbf{W}^\top\boldsymbol{\phi}(\mathbf{x})=\boldsymbol{\Psi}^\top\boldsymbol{\Phi}\boldsymbol{\phi}(\mathbf{x})=\boldsymbol{\Psi}^\top\mathbf{k}(\mathbf{x})\\&=\left(\mathrm{mat}\left(\mathbf{B}\mathrm{vec}(\mathbf{T}_D)\right)\right)^\top\mathbf{k}(\mathbf{x}),\end{aligned}\tag{12}$$

where $\mathbf{k}(\mathbf{x})=[k_1(\mathbf{x}),k_2(\mathbf{x}),\cdots,k_N(\mathbf{x})]^\top$. Here $\mathrm{mat}(\cdot)$ denotes the reshaping operation of an argument vector into a matrix of size $N\times M$ by concatenating subsequent N length sections as columns. The complete process of KRGS is summarized in Algorithm 1.

Algorithm 1 Kernel Regression over Graphs for Sparse Noise

1: Initialize $\mathbf{D}_n=\mathbf{I}_M$ for all $1\le n\le N$, set i_{max} maximum number of iterations
2: **while** $i<i_{max}$ **do**
3: Compute $\boldsymbol{\Psi}=\mathbf{B}\mathrm{vec}(\mathbf{T}_D)$ where $\mathbf{B}$ is as given in (11).
4: For input $\mathbf{x}$, output predicted $\mathbf{y}=\boldsymbol{\Psi}^\top\mathbf{k}(\mathbf{x})$.
5: $\mathbf{D}_n=\mathrm{diag}(d_n(1),d_n(2),\cdots,d_n(M))$ where $d_n(m)$ is given by (4).
6: $i\to i+1$

3. EXPERIMENTS

We consider the application of our approach to real-world graph signal of temperature measurements taken over 45 most populated cities in Sweden for the period of three months from September to November 2017 [1]. Our hypothesis is that our approach achieves good prediction performance in the two cases of corruptions in the graph signal training data: missing samples, and large perturbations.

The input $\mathbf{x}\in\mathbb{R}^{45}$ is the vector of temperature measurements of all 45 cities of one day, and the output $\mathbf{t}_n\in\mathbb{R}^{45}$ to be predicted contains the temperature measurements of the 45 cities for the succeeding day. This gives us a total of 92 input-output pairs. We use the first

[1]The data is available publicly from the Swedish Meteorological and Hydrological Institute [21].

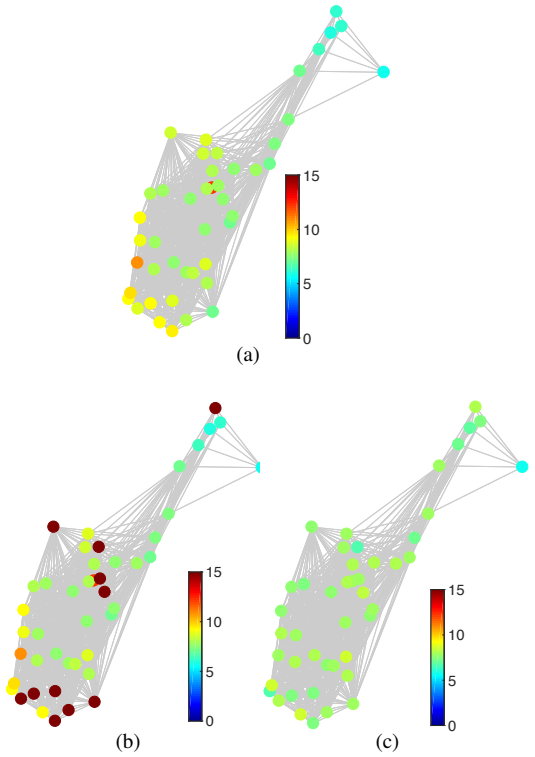

Fig. 1. Example showing result of our approach on temperature data over Sweden on training data: (a) True output (b) Output with sparse noise, and (c) Predicted output from our approach. The colorbar shows the temperature scale in degree celsius.

$N_{tr} = 46$ pairs for training, and the remaining $N_t = 46$ for testing. We consider the graph adjacency matrix constructed using geodesic distances between the cities as $\mathbf{A}(i,j) = \exp\left(-\frac{d_{ij}^2}{\sum_{i,j} d_{ij}^2}\right)$, where $d_{i,j}$ denotes the geodesic distance between the ith and jth cities. We measure the prediction performance on test data in terms of normalized mean square error (NMSE):

$$\text{NMSE} = 10\log_{10}\left(\frac{\text{E}\left(\sum_{n=1}^{Nt}\|\mathbf{y}_n - \mathbf{t}_{0,n}\|_2^2\right)}{\text{E}(\sum_{n=1}^{Nt}\|\mathbf{t}_{0,n}\|_2^2)}\right),$$

where $\mathbf{y}_n$ denotes the regression output and $\mathbf{t}_{0,n}$ the true value of output which does not contain any noise, and Nt denotes the number of test data points. The experiment is performed for different training data sizes of N drawn from the full training set of $N_{tr} = 46$ samples. The parameter δ is set experimentally to 0.1. The parameters α and β are computed through four-fold cross validation. We use Gaussian kernel in all our experiments whose parameter σ is also set using four-fold cross-validation. We initialize our algorithm with $\mathbf{D}_n$ as the identity matrix for all N samples. This corresponds to solving the cost with ℓ_2 norm of the model error as the first iteration. The regression problem with such a cost was defined in [12], where it was termed as the kernel regression over graphs (KRG). We compare the performance to the baseline performance obtained using KRG as the iterations increase. This gives us a clear picture of the gain in using KRGS over KRG. We note here that KRG has been shown to outperform kernel ridge regression (KRR) [5], a state-of-the-art approach in graph signal reconstruction which employs kernels across nodes [12]. Since KRR was reported to outperform other competing approaches, it suffices to make comparisons only with KRG. Although other non-kernel-based approaches specific to temporal graph signal prediction exist [22–24], we make comparisons with KRG to demonstrate the performance in the context of kernel-based techniques.

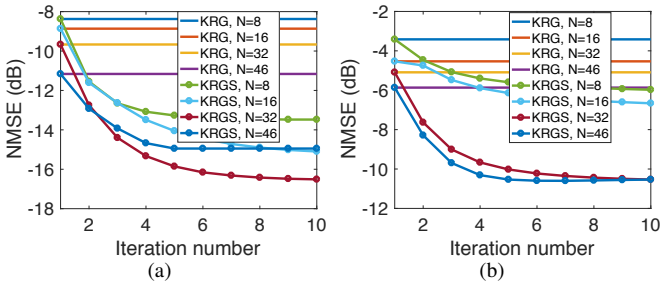

Fig. 2. Results for temperature prediction experiment. NMSE as a function of number of iterations, at different training sample sizes for (a) missing data, and (b) Large perturbations.

We simulate the missing samples as follows: For each training output graph signal, we randomly choose 25% of the nodes and set their signal values to equal to zero. The subset of nodes where the signal values are missing is independent from one training observation to the other. In the case of large perturbations, we adopt the same strategy but instead of setting signal values to zero, we scale them by a constant factor of 4. This experimentally results in a training samples to have a signal-to-noise-ratio (SNR) of in the range of 6 to 10dB in both the sparse noise cases. An example of application of our approach with large perturbations is illustrated in Figure 1. In Figure 2, we show the NMSE as a function of the iterations for various training sample sizes N, averaged over 100 Monte Carlo simulations of missing samples or perturbations. We observe that for each training sample size N, the NMSE reduces with the number of iterations, and we find that convergence is acheived after five to ten iterations. We also observe that our approach yields significant gain in prediction performance compared to KRG, which corresponds to the solution obtained at the first iteration of our approach. We also note that while we have made use of a single δ in all our experiments, changing δ with N could possibly help achieve even better prediction performance.

4. CONCLUSIONS

We proposed linear regression for predicting graph signals for the adverse scenario when the training data is corrupted with sparse noise, by posing it as a regularized ℓ_1-norm minimization problem. We employed the iteratively reweighted least-squares approach to solve the optimization problem, which resulted in a closed form solution. This in turn enabled the development of the more general kernel regression for sparse noise scenario, through the use of kernel substitution. Application on a real-world graph signal dataset showed that our approach outperforms the ℓ_2-norm based kernel regression by a very significant margin, and that the performance improves with the number of iterations.

A detailed analysis of the convergence properties and performance evaluation on different datasets under different noise and hyperparameter settings would be the subject of our future work.

5. REFERENCES

[1] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[2] G.-B. Huang, X. Ding, and H. Zhou, "Optimization method based extreme learning machine for classification," *Neurocomputing*, vol. 74, no. 1, pp. 155 – 163, 2010.

[3] Y. Cho and L. K. Saul, "Kernel methods for deep learning," in *Adv. Neural Inform. Process. Syst.* Curran Associates, Inc., 2009, pp. 342–350.

[4] A. J. Smola and R. Kondor, *Kernels and Regularization on Graphs*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, pp. 144–158.

[5] 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.

[6] ——, "Estimating signals over graphs via multi-kernel learning," in *IEEE Statist. Signal Process. Workshop (SSP)*, June 2016, pp. 1–5.

[7] V. N. Ioannidis, D. Romero, and G. B. Giannakis, "Kernel-based reconstruction of space-time functions via extended graphs," in *Asilomar Conf. Signals Syst. Comput.*, Nov 2016, pp. 1829–1833.

[8] D. Romero, V. N. Ioannidis, and G. B. Giannakis, "Kernel-based reconstruction of space-time functions on dynamic graphs," *CoRR*, vol. abs/1612.03615, 2016. [Online]. Available: http://arxiv.org/abs/1612.03615

[9] V. N. Ioannidis, M. Ma, A. N. Nikolakopoulos, G. B. Giannakis, and D. Romero, "Kernel-based inference of functions over graphs," in *Adaptive Learning Methods for Nonlinear System Modeling*, D. Comminiello and J. C. Príncipe, Eds. Butterworth-Heinemann, 2018, pp. 173 – 198.

[10] M. K. Chung, A. Qiu, S. Seo, and H. K. Vorperian, "Unified heat kernel regression for diffusion, kernel smoothing and wavelets on manifolds and its application to mandible growth modeling in ct images," *Medical Image Analysis*, vol. 22, no. 1, pp. 63 – 76, 2015.

[11] Y. Shen, B. Baingana, and G. B. Giannakis, "Kernel-based structural equation models for topology identification of directed networks," *IEEE Trans. Signal Process.*, vol. 65, no. 10, pp. 2503–2516, May 2017.

[12] A. Venkitaraman, S. Chatterjee, and P. Händel, "Kernel Regression for Signals over Graphs," *ArXiv e-prints*, Jun. 2017.

[13] A. Venkitaraman, S. Chatterjee, and P. Händel, "Gaussian processes over graphs," *https://arxiv.org/abs/1803.05776*, 2018.

[14] D. Romero, M. Ma, and G. B. Giannakis, "Estimating signals over graphs via multi-kernel learning," in *IEEE Statistical Signal Processing Workshop (SSP)*, June, pp. 1–5.

[15] A. Venkitaraman, S. Chatterjee, and P. Handel, "Multi-kernel regression for graph signal processing," in *Int. Conf. Acoust. Speech Signal Process. Calgary, Canada*, 2018, pp. 4644–4648. [Online]. Available: https://doi.org/10.1109/ICASSP.2018.8461643

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

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

[18] I. Daubechies, R. DeVore, M. Fornasier, and C. S. Güntürk, "Iteratively reweighted least squares minimization for sparse recovery," *Communications on Pure and Applied Mathematics*, vol. 63, no. 1, pp. 1–38.

[19] R. Chartrand and W. Yin, "Iteratively reweighted algorithms for compressive sensing," in *IEEE Intl. Conf. Acoust. Speech Signal Process.*, March 2008, pp. 3869–3872.

[20] C. F. V. Loan, "The ubiquitous Kronecker product," *J. Computat. Appl. Mathematics*, vol. 123, no. 1–2, pp. 85–100, 2000.

[21] Swedish meteorological and hydrological institute (smhi). [Online]. Available: http://opendata-download-metobs.smhi.se/

[22] J. Mei and J. M. F. Moura, "Signal processing on graphs: Causal modeling of unstructured data," *IEEE Trans. Signal Process.*, vol. 65, no. 8, pp. 2077–2092, April 2017.

[23] K. Qiu, X. Mao, X. Shen, X. Wang, T. Li, and Y. Gu, "Time-varying graph signal reconstruction," *IEEE J. Selected Topics Signal Process.*, vol. 11, no. 6, pp. 870–883, Sep. 2017.

[24] E. Isufi, A. Loukas, N. Perraudin, and G. Leus, "Forecasting time series with varma recursions on graphs," *eprint arXiv:1810.08581*, 2018.