Extreme Learning Machine for Graph Signal Processing

Arun Venkitaraman, Saikat Chatterjee, Peter Händel
Department of Information Science and Engineering
School of Electrical Engineering, KTH Royal Institute of Technology, Stockholm, Sweden
arunv@kth.se, sach@kth.se, ph@kth.se

***Abstract*—In this article, we improve extreme learning machines for regression tasks using a graph signal processing based regularization. We assume that the target signal for prediction or regression is a graph signal. With this assumption, we use the regularization to enforce that the output of an extreme learning machine is smooth over a given graph. Simulation results with real data confirm that such regularization helps significantly when the available training data is limited in size and corrupted by noise.**

I. Introduction

Extreme learning machines (ELMs) have emerged as an active area of research within the machine learning community [1]. ELMs differ from the traditional approaches such as neural networks (NNs) and suppor vector machines (SVMs) in an important respect: the parameters of hidden nodes of the ELM are randomly generated and the learning takes place only at the output layer or at the extreme layer by solving a regularized least-squares problem [2]. As a result, ELM does not suffer from computational issues that often affect traditional approaches, making the ELM a fast and effective learning approach. ELM despite its simplicity has been shown to have high quality performances in classification and regression tasks, often giving similar or better results in comparison with NNs and SVMs, while being orders of magnitude faster[2], [3], [4], [5]. In fact, the ELM enjoys universal approximation properties: given mild conditions on the activation functions, the ELM can be shown to approximate any continuous(in the case of regression) or piecewise continuous (in the case of classification) target function as the number of neurons in the hidden layer tends to infinity [6], [1]. Though traditionally ELMs have been developed as single-layer feed forward networks, multi-layer, distributed, and incremental extensions have also been developed [7], [8], [9].

As with any machine learning paradigm, the performance of the ELM depends on the nature of the training and testing data. The more abundant and reliable the training data be, the better the classification or regression performance be. However, in many applications the training data may be scarce and corrupted by noise. In such cases, it is important to incorporate additional structures during the training phase. In this article, we propose the use of the emerging notion of graph signal processing [10], [11] to enhance the prediction performance of the ELM in regression tasks. In particular, we consider the target or output of the ELM to be smooth signals over a graph, and propose ELM for graph signal processing (ELMG). We note that graph-based regularizations have been successfully employed in literature in different contexts such as labelling and kernel regression[12], [13], [14], [15], [16]. Our hypothesis is that such an approach results in improved prediction performance. Experiments with real-world data show the validity of our hypothesis. ELMs and graph signal processing have both emerged as directions of much interest in the respective communities, and we believe that our work is a step towards bringing them together.

II. Preliminaries

A. Graph signal processing

Consider a graph of M nodes denoted by $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{A})$ where $\mathcal{V}$ denotes the node set, $\mathcal{E}$ the edge set, and $\mathbf{A} = [a_{ij}]$ the adjacency matrix, $a_{ij} \geq 0$. Since we consider only undirected graphs, we have that $\mathbf{A}$ is symmetric [17]. A vector $\mathbf{y} = [y(1)y(2)\cdots y(M)]^\top \in \mathbb{R}^M$ is said to be a graph signal over $\mathcal{G}$ if $y(m)$ denotes the value of the signal at the mth node of the graph [10], [11], [18], [19], [20], [21]. The smoothness of a graph signal $\mathbf{y}$ is measured in terms of the quadratic form:

$$l(\mathbf{y}) = \mathbf{y}^\top \mathbf{L}\mathbf{y} = \sum_{(i,j)\in\mathcal{E}} a_{ij}(y(i) - y(j))^2,$$

where $\mathbf{L} = \mathbf{D} - \mathbf{A}$ is the graph Laplacian matrix, $\mathbf{D}$ being the diagonal degree matrix with ith diagonal given by $d_i = \sum_j a_{ij}$. $l(\mathbf{y})$ is a measure of variation of $\mathbf{y}$ across connected nodes: the smaller the value of $l(\mathbf{y})$ implies the smoother the signal $\mathbf{y}$.

B. Extreme learning machine

Consider a set of N observations of input and target pairs $\{(\mathbf{x}_n, \mathbf{y}_n)\}_{n=1}^N$. An extreme learning machine consists of the input layer, an output layer and a hidden layer of K neurons as shown in Figure 1. The kth neuron implements a nonlinear operation $h_k(\cdot)$ on the input and is parameterized by set of variables $(\mathbf{a}_k, \mathbf{b}_k)$, that is,

$$h_k(\mathbf{x}) = G(\mathbf{x}, \mathbf{a}_k, \mathbf{b}_k)$$

where $G(\cdot, \mathbf{a}, \mathbf{b})$ is a parametric scalar function, for example, the sigmoid function $G(\mathbf{x}, \mathbf{a}, b) = \dfrac{1}{1 + \exp\left(-(\mathbf{a}^\top \mathbf{x} + b)\right)}$. The functions $h_k(\cdot)$ are referred to as the activation functions.

ISBN 978-90-827970-1-5

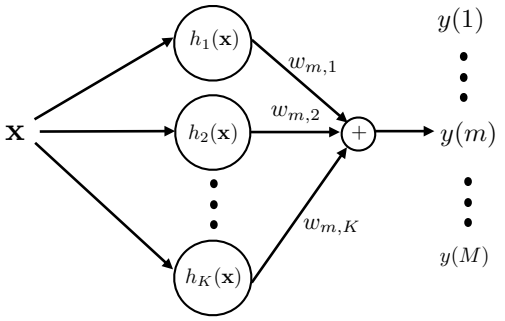

Fig. 1. Schematic of an extreme learning machine for vector target.

The parameters $(\mathbf{a}_k, \mathbf{b}_k)$ are drawn randomly from a known probability distribution. The weights $w_{m,k}$ corresponds to the regression coefficient relating the output of the kth neuron to the mth component of the vector output $\mathbf{y} \in \mathbb{R}^M$. ELM models the output or target vector $\mathbf{y}$ as the output of linear regression such that

$$\mathbf{y} = \mathbf{W}^\top \mathbf{h}(\mathbf{x}) \tag{1}$$

where $\mathbf{h}(\mathbf{x}) = [h_1(\mathbf{x}), h_2(\mathbf{x}), \cdots, h_K(\mathbf{x})]^\top$. In the ELM, we learn the regression matrix $\mathbf{W}$ by solving the following regularized least-squares problem:

$$\arg\min_{\mathbf{W}} \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{W}^\top \mathbf{h}(\mathbf{x}_n)\|_2^2 + \alpha \text{tr}(\mathbf{W}^\top \mathbf{W}),\ \alpha \geq 0 \tag{2}$$

where tr$(\cdot)$ denotes the matrix trace operation. Let us define the matrices $\mathbf{T}$ and $\mathbf{H}$ such that $\mathbf{T} = [\mathbf{t}_1 \mathbf{t}_2 \cdots \mathbf{t}_N]^\top$ and $\mathbf{H} = [\mathbf{h}(\mathbf{x}_1)\mathbf{h}(\mathbf{x}_2) \cdots \mathbf{h}(\mathbf{x}_N)]^\top$. Then, we have that

$$\mathbf{W} = \arg\min_{\mathbf{W}} \left(\|\mathbf{T} - \mathbf{H}\mathbf{W}\|_F^2 + \alpha \text{tr}(\mathbf{W}^\top \mathbf{W}) \right),\ \alpha \geq 0$$

where $\|\cdot\|_F$ denotes the Frobenius norm. The optimal value of $\mathbf{W}$ is obtained as [6]:

$$\mathbf{W} = (\mathbf{H}^\top \mathbf{H} + \alpha \mathbf{I}_N)^{-1} \mathbf{T}. \tag{3}$$

Thus, ELM consists of input layer acted upon by activation functions with random parameters, followed by learning of the regression coefficients at the output layer.

III. ELM OVER GRAPH

We next propose ELM for graph signal processing. We assume that the target signal $\mathbf{t}$ is a graph signal and may be corrupted by noise. Our goal is to learn an ELM such that the predicted output $\mathbf{y}$ is smooth over graph $\mathcal{G}$. We achieve this goal by enforcing a penalty of the form $\sum_n l(\mathbf{y}_n)$ while training the ELM. In other words, we learn the optimal regression coefficients as the minimizer of the following optimization problem:

$$C(\mathbf{W}) = \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \text{tr}(\mathbf{W}^\top \mathbf{W}) + \beta \sum_{n=1}^{N} l(\mathbf{y}_n),$$
$$\text{subject to } \mathbf{y}_n = \mathbf{W}^\top \mathbf{h}(\mathbf{x}_n)\ \forall\, n, \quad \alpha, \beta \geq 0. \tag{4}$$

We observe that (4) is an optimization similar to that in (2) and therefore represents an ELM with further regularization: the ELM output is enforced to be smooth over the graph $\mathcal{G}$. Then, the predicted output for a new input $\mathbf{x}$ is given by

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}_g^\top \mathbf{h}(\mathbf{x}),$$

which we refer to as the output of the ELMG. We next proceed to evaluate the optimal $\mathbf{W}_g$ matrix. Using properties of the matrix trace operation trace$(\cdot)$, we have that

$$\begin{aligned}
C(\mathbf{W}) &= \|\mathbf{T} - \mathbf{H}\mathbf{W}\|_F^2 + \alpha \text{tr}(\mathbf{W}^\top \mathbf{W}) \\
&\quad + \beta \sum_n \mathbf{h}(\mathbf{x}_n)^\top \mathbf{W}\mathbf{L}\mathbf{W}^\top \mathbf{h}(\mathbf{x}_n) \\
&= \text{tr}((\mathbf{T} - \mathbf{H}\mathbf{W})^\top (\mathbf{T} - \mathbf{H}\mathbf{W})) + \alpha \text{tr}(\mathbf{W}^\top \mathbf{W}) \\
&\quad + \beta \sum_n \mathbf{h}(\mathbf{x}_n)^\top \mathbf{W}\mathbf{L}\mathbf{W}^\top \mathbf{h}(\mathbf{x}_n) \\
&= \text{tr}(\mathbf{T}^\top \mathbf{T}) - 2\,\text{tr}\left(\mathbf{T}^\top \mathbf{H}\mathbf{W}\right) \\
&\quad + \text{tr}(\mathbf{W}^\top \mathbf{H}^\top \mathbf{H}\mathbf{W}) + \alpha\, \text{tr}(\mathbf{W}^\top \mathbf{W}) \\
&\quad + \beta\, \text{tr}\left(\sum_n \mathbf{h}(\mathbf{x}_n)^\top \mathbf{W}\mathbf{L}\mathbf{W}^\top \mathbf{h}(\mathbf{x}_n) \right) \\
&= \text{tr}(\mathbf{T}^\top \mathbf{T}) - 2\,\text{tr}\left(\mathbf{T}^\top \mathbf{H}\mathbf{W}\right) + \text{tr}(\mathbf{W}^\top (\mathbf{H}^\top \mathbf{H} + \alpha \mathbf{I})\mathbf{W}) \\
&\quad + \beta\, \text{tr}\left(\mathbf{W}\mathbf{L}\mathbf{W}^\top \sum_n \mathbf{h}(\mathbf{x}_n)\mathbf{h}(\mathbf{x}_n)^\top \right) \\
&= \text{tr}(\mathbf{T}^\top \mathbf{T}) - 2\,\text{tr}\left(\mathbf{T}^\top \mathbf{H}\mathbf{W}\right) + \text{tr}(\mathbf{W}^\top \mathbf{H}^\top \mathbf{H}\mathbf{W}) \\
&\quad + \alpha\, \text{tr}(\mathbf{W}^\top \mathbf{W}) + \beta\, \text{tr}\left(\mathbf{W}^\top \mathbf{H}^\top \mathbf{H}\mathbf{W}\mathbf{L}\right).
\end{aligned} \tag{5}$$

$C(\mathbf{W})$ is quadratic in $\mathbf{W}$. Hence, we obtain the optimal and unique solution by setting the gradient of C with respect to $\mathbf{W}$ equal to zero. Using matrix derivative relations[22]

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{W}} \text{tr}\left(\mathbf{M}_1 \mathbf{W}\right) &= \mathbf{M}_1^\top, \\
\frac{\partial}{\partial \mathbf{W}} \text{tr}\left(\mathbf{W}^\top \mathbf{M}_1 \mathbf{W} \mathbf{M}_2\right) &= \mathbf{M}_1^\top \mathbf{W} \mathbf{M}_2^\top + \mathbf{M}_1 \mathbf{W} \mathbf{M}_2,
\end{aligned}$$

where $\mathbf{M}_1$ and $\mathbf{M}_2$ are matrices, and setting $\frac{\partial C}{\partial \mathbf{W}} = 0$ we get that

$$\begin{aligned}
&-\mathbf{H}^\top \mathbf{T} + \mathbf{H}^\top \mathbf{H}\mathbf{W} + \alpha \mathbf{W} + \beta \mathbf{H}^\top \mathbf{H}\mathbf{W}\mathbf{L} = \mathbf{0}, \\
\text{or,}\quad &\left(\mathbf{H}^\top \mathbf{H} + \alpha \mathbf{I}_K\right) \mathbf{W} + \beta \mathbf{H}^\top \mathbf{H}\mathbf{W}\mathbf{L} = \mathbf{H}^\top \mathbf{T}.
\end{aligned} \tag{6}$$

On rearranging and vectorizing terms in (6), we have that

$$\text{vec}(\mathbf{H}^\top \mathbf{T}) = \left[(\mathbf{I}_M \otimes (\mathbf{H}^\top \mathbf{H} + \alpha \mathbf{I}_K)) + (\beta \mathbf{L} \otimes \mathbf{H}^\top \mathbf{H})\right] \text{vec}(\mathbf{W}),$$

ISBN 978-90-827970-1-5 © EURASIP 2018

where vec$(\cdot)$ denotes the standard vectorization operator and $\otimes$ denotes the Kronecker product operation [23]. Then, $\mathbf{W}_g$ follows the relation:

$$\begin{aligned}&\text{vec}(\mathbf{W}_g)\\&=\left[(\mathbf{I}_M\otimes(\mathbf{H}^\top\mathbf{H}+\alpha\mathbf{I}_K))+(\beta\mathbf{L}\otimes\mathbf{H}^\top\mathbf{H})\right]^{-1}\text{vec}(\mathbf{H}^\top\mathbf{T})\\&=\mathbf{G}^{-1}(\mathbf{I}\otimes\mathbf{H}^\top)\text{vec}(\mathbf{T})\end{aligned}\tag{7}$$

where $\mathbf{G}=\mathbf{I}_M\otimes(\mathbf{H}^\top\mathbf{H}+\alpha\mathbf{I})+\beta\mathbf{L}\otimes\mathbf{H}^\top\mathbf{H}$. We observe that on setting $\beta=0$ or $\mathbf{L}=\mathbf{0}$ which corresponds to having no graph, we get that

$$\begin{aligned}\text{vec}(\mathbf{W}_g)&=\left[\mathbf{I}_M\otimes(\mathbf{H}^\top\mathbf{H}+\alpha\mathbf{I}_K)\right]^{-1}\text{vec}(\mathbf{H}^\top\mathbf{T}),\ \text{or}\\\mathbf{W}_g&=(\mathbf{H}^\top\mathbf{H}+\alpha\mathbf{I}_N)^{-1}\mathbf{T}.\end{aligned}$$

In other words, ELMG reduces to the standard ELM when $\beta=0$.

IV. Smoothing effects for ELMG

We next discuss how the ELMG output for training data can be interpreted as a smoothing action across both observations and graph nodes. We show that the ELMG performs a shrinkage along the principal components of the graph Laplacian and kernel matrix. On vectorizing $\mathbf{Y}=\mathbf{H}\mathbf{W}_g$ and using (7), we get that

$$\begin{aligned}\text{vec}(\mathbf{Y})&=(\mathbf{I}_M\otimes\mathbf{H})\text{vec}(\mathbf{W}_g)\\&=(\mathbf{I}_M\otimes\mathbf{H})\mathbf{G}^{-1}(\mathbf{I}_M\otimes\mathbf{H}^\top)\text{vec}(\mathbf{T}).\end{aligned}\tag{8}$$

Let $\mathbf{L}$ be diagonalizable with the eigendecomposition:

$$\mathbf{L}=\mathbf{V}\mathbf{J}_L\mathbf{V}^\top,$$

where $\mathbf{J}_L$ and $\mathbf{V}$ denote the eigenvalue and eigenvector matrices of $\mathbf{L}$, respectively, such that

$$\begin{aligned}\mathbf{V}&=[\mathbf{v}_1\,\mathbf{v}_2\cdots\mathbf{v}_N]\in\mathbb{R}^{M\times M}\\\mathbf{J}_L&=\text{diag}(\lambda_1,\lambda_2,\cdots,\lambda_N)\in\mathbb{R}^{M\times M}.\end{aligned}$$

Further, let $\mathbf{H}=\mathbf{U}\mathbf{\Sigma}\mathbf{V}_H^\top$ denote the reduced singular-value decomposition of $\mathbf{H}$ such that

$$\begin{aligned}\mathbf{U}_H&=[\mathbf{u}_{H,1}\,\mathbf{u}_{H,2}\cdots\mathbf{u}_{H,M}]\in\mathbb{R}^{M\times r}\\\mathbf{\Sigma}&=\text{diag}(\sigma_1^2,\sigma_2^2,\cdots,\sigma_N^2)\in\mathbb{R}^{r\times r}\\\mathbf{V}_H&=[\mathbf{v}_{H,1}\,\mathbf{v}_{H,2}\cdots\mathbf{v}_{H,r}]\ \in\mathbb{R}^{r\times N}\text{and}\end{aligned}$$

where r denotes the rank of $\mathbf{H}$ equal to the number of nonzero singular values, and $\mathbf{\Sigma}$ is the reduced singular value matrix which is the submatrix with only non-zero diagonal of the full singular value matrix. Then, we have that

$$\begin{aligned}\mathbf{G}&=\mathbf{I}_M\otimes(\mathbf{H}^\top\mathbf{H}+\alpha\mathbf{I})+\beta\mathbf{L}\otimes\mathbf{H}^\top\mathbf{H}\\&=[(\mathbf{V}\mathbf{I}_M\mathbf{V}^\top)\otimes(\mathbf{V}_H(\mathbf{\Sigma}^2+\alpha\mathbf{I}_N)\mathbf{V}_H^\top)]\\&\quad+[\beta(\mathbf{V}\mathbf{J}_L\mathbf{V}^\top)\otimes(\mathbf{V}_H\mathbf{\Sigma}^2\mathbf{V}_H^\top)]\\&\overset{(a)}{=}[(\mathbf{V}\otimes\mathbf{V}_H)(\mathbf{I}_M\otimes(\mathbf{\Sigma}^2+\alpha\mathbf{I}_N))(\mathbf{V}^\top\otimes\mathbf{V}_H^\top)]\\&\quad+[\beta(\mathbf{V}\otimes\mathbf{V}_H)(\mathbf{J}_L\otimes\mathbf{\Sigma}^2)(\mathbf{V}^\top\otimes\mathbf{V}_H^\top)]\\&=(\mathbf{V}\otimes\mathbf{V}_H)\mathbf{J}(\mathbf{V}\otimes\mathbf{V}_H)^\top,\end{aligned}\tag{9}$$

where $\mathbf{J}=(\mathbf{I}_M\otimes(\mathbf{\Sigma}^2+\alpha\mathbf{I}_N))+\beta(\mathbf{J}_L\otimes\mathbf{\Sigma}^2)\in\mathbb{R}^{Mr\times Mr}$. In (9)(a), we have used the distributivity of the Kronecker product: $(\mathbf{M}_1\otimes\mathbf{M}_2)(\mathbf{M}_3\otimes\mathbf{M}_4)=\mathbf{M}_1\mathbf{M}_3\otimes\mathbf{M}_2\mathbf{M}_4$ where $\{\mathbf{M}_i\}_{i=1}^4$ are four matrices. Similarly, we have that

$$\begin{aligned}(\mathbf{I}_M\otimes\mathbf{H})&=(\mathbf{V}\mathbf{I}_M\mathbf{V}^\top)\otimes(\mathbf{U}_H\mathbf{\Sigma}\mathbf{V}_H^\top)\\&=(\mathbf{V}\otimes\mathbf{U}_H)(\mathbf{I}_M\otimes\mathbf{\Sigma})(\mathbf{V}^\top\otimes\mathbf{V}_H^\top)\end{aligned}\tag{10}$$

Similarly,

$$(\mathbf{I}_M\otimes\mathbf{H}^\top)=(\mathbf{V}\otimes\mathbf{V}_H)(\mathbf{I}_M\otimes\mathbf{\Sigma})(\mathbf{V}^\top\otimes\mathbf{U}_H^\top).\tag{11}$$

Then, on substituting (9), (10), and (11) in (8), we get that

$$\begin{aligned}&\text{vec}(\mathbf{Y})\\&=(\mathbf{Z}(\mathbf{I}_M\otimes\mathbf{\Sigma})\mathbf{J}^{-1}(\mathbf{I}_M\otimes\mathbf{\Sigma})\mathbf{Z}^\top)\text{vec}(\mathbf{T})\\&=(\mathbf{Z}[(\mathbf{I}_M\otimes(\mathbf{I}+\alpha\mathbf{\Sigma}^{-2}))+\beta(\mathbf{J}_L\otimes\mathbf{I}))]^{-1}\mathbf{Z}^\top)\text{vec}(\mathbf{T}),\end{aligned}\tag{12}$$

where $\mathbf{Z}=\mathbf{V}\otimes\mathbf{U}_H$.

Let $\mathbf{J}_F=[(\mathbf{I}_M\otimes(\mathbf{I}+\alpha\mathbf{\Sigma}^{-2}))+\beta(\mathbf{J}_L\otimes\mathbf{I}))]^{-1}$. Then, any diaognal element ζ_i of $\mathbf{J}_F$ is of the form

$$\zeta_i=[(1+\alpha\sigma_{i2}^{-2})+\beta\lambda_{i1}]^{-1}=\frac{1}{[(1+\beta\lambda_{i1})+\alpha/\sigma_{i2}^2]},$$

for some $i1\leq M$ and $i2\leq N$. From (12), we have that

$$\text{vec}\left(\mathbf{Y}\right)=\sum_{i=1}^{MN}\zeta_i\mathbf{z}_i\mathbf{z}_i^\top\,\text{vec}\left(\mathbf{T}\right),\tag{13}$$

where $\mathbf{z}_i$ are column vectors of $\mathbf{Z}$. (13) expresses the prediction output $\mathbf{Y}$ as projections along the principal directions given by $\mathbf{z}_i=\mathbf{v}_{i1}\otimes\mathbf{u}_{H,i2}$ for some $i1\leq M$ and $i2\leq N$. In the case when $\zeta_i\ll 1$, the component in vec$(\mathbf{T})$ along $\mathbf{z}_i$ is effectively eliminated. The principal components corresponding to largest σ^2 correspond to the most informative directions. The components corresponding to smaller σ^2 are usually those of noise with high-frequency components. The eigenvectors of $\mathbf{L}$ corresponding to the smaller eigenvalues λ are smooth over the graph [17], [10]. We observe that the condition $\zeta_i\ll 1$ is achieved when corresponding σ_{i1}^2 is small and/ or λ_{i2} is large. This corresponds to effectively retaining only components in vec$(\mathbf{T})$ which vary smoothly across the observation inputs $\{1,\cdots,N\}$ and and/ or smoothly varying across over the M nodes of the graph. The extend of the smoothing achieved depends on the regularization parameters α and β. Thus, ELMG output corresponds to a smoothing or denoising operation over the training targets.

V. Experiments

We consider the application of the proposed ELM over graphs to two real-world graph signal datasets. In each of these datasets, the true targets $\mathbf{t}_{o,n}$'s are smooth graph signals which lie over a specified graph. We assume that the true target values are not observed and that we have access to only noisy target $\mathbf{t}_n$'s and the corresponding inputs $\mathbf{x}_n$'s. In order to simulate such a situation, we deliberately corrupt the true graph signal targets with additive white Gaussian noise $\mathbf{e}_n$:

$$\mathbf{t}_n=\mathbf{t}_{o,n}+\mathbf{e}_n.$$

We then use the noisy targets for training ELM and ELMG. The trained models are then used to predict targets for inputs

ISBN 978-90-827970-1-5 © EURASIP 2018

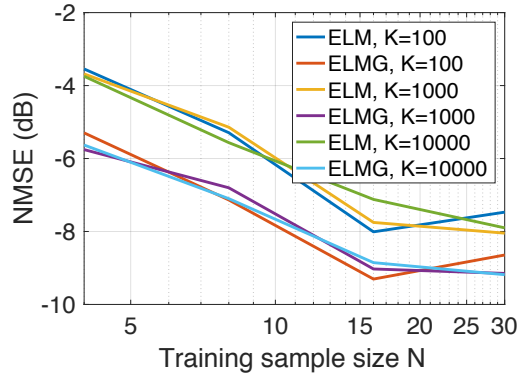

Fig. 2. NMSE for temperature measurements with sigmoid activation function.

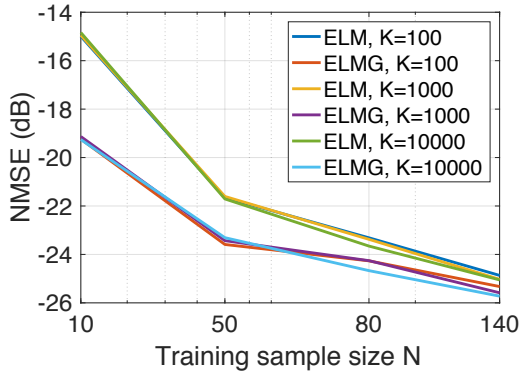

Fig. 3. NMSE on for fMRI measurements with sigmoid activation function.

K	ELM	ELMG	ELM	ELMG	ELM	ELMG
	$N=4$		$N=16$		$N=30$	
Sigmoid function						
10^2	-3.74	**-6.58**	-7.72	**-9.36**	-7.87	**-8.99**
10^3	-3.78	**-6.56**	-7.53	**-9.05**	-7.74	**-8.91**
10^4	-3.94	**-6.71**	-7.63	**-9.24**	-8.02	**-9.10**
Hardlimit function						
10^2	-3.55	**-7.14**	-8.01	**-9.30**	-7.47	**-8.65**
10^3	-3.68	**-6.80**	-7.75	**-9.03**	-8.05	**-9.15**
10^4	-3.75	**-7.10**	-7.12	**-8.85**	-7.90	**-9.18**
Gaussian function						
10^2	-3.73	**-6.34**	-7.61	**-9.11**	-7.77	**-8.92**
10^3	-3.66	**-6.38**	-7.55	**-9.17**	-7.90	**-9.02**
10^4	-3.70	**-6.69**	-7.58	**-9.09**	-7.80	**-9.00**

TABLE I
NMSE FOR TESTING DATA AS A FUNCTION OF K AND N FOR TEMPERATURE MEASUREMENTS.

K	ELM	ELMG	ELM	ELMG	ELM	ELMG
	$N=10$		$N=80$		$N=140$	
Sigmoid function						
10^2	-14.99	**-19.24**	-23.31	**-24.26**	-24.86	**-25.33**
10^3	-14.95	**-19.13**	-23.38	**-24.26**	-25.03	**-25.58**
10^4	-14.84	**-19.26**	-23.66	**-24.67**	-25.05	**-25.72**
Hardlimit function						
10^2	-14.99	**-19.24**	-23.31	**-24.26**	-24.87	**-25.33**
10^3	-14.95	**-19.13**	-23.38	**-24.26**	-25.03	**-25.58**
10^4	-14.84	**-19.26**	-23.66	**-24.67**	-25.05	**-25.72**
Gaussian function						
10^2	-15.06	**-19.05**	-23.55	**-24.62**	-25.68	**-26.14**
10^3	-15.02	**-19.12**	-23.49	**-24.57**	-25.66	**-26.24**
10^4	-15.09	**-19.13**	-23.56	**-24.63**	-25.75	**-26.30**

TABLE II
NMSE FOR TEST SET AS A FUNCTION OF K AND N FOR CEREBELLUM FMRI DATA.

from the test dataset. We consider the following three different activation functions popular in ELM literature:

1) Sigmoid function: $G(\mathbf{x},\mathbf{a},b) = \dfrac{1}{1+\exp(-(\mathbf{a}^\top\mathbf{x}+b))}$.
2) Hardlimit function: $G(\mathbf{x},\mathbf{a},b) = \begin{cases} 1, & \text{if } \mathbf{a}^\top\mathbf{x}+b \geq 0, \\ 0, & \text{otherwise.} \end{cases}$
3) Gaussian function: $G(\mathbf{x},\mathbf{a},b) = \exp(-b\|\mathbf{x}-\mathbf{a}\|_2^2)$.

The entries of parameters $\mathbf{a}$ and b for each of the K neurons are drawn independently from the standard normal distribution. The parameters α and β are found by exhaustive grid search. We compare the prediction performance of both the strategies in terms of the normalized mean-square error (NMSE) for the test data, averaged over 100 different dataset partitions and noise realizations. As discussed earlier, our hypothesis is that graph signal structure helps to improve prediction by the ELM. Our experiments below show that this is indeed the case. The codes used for experiments may be found at: https://www.kth.se/en/ees/omskolan/organisation/avdelningar/information-science-and-engineering/research/reproducibleresearch.

A. Temperature of cities in Sweden

We consider the temperature measurements over the 45 largest cities in Sweden for the period of October to December 2015. We consider the geodesic graph whose adjacency matrix is given by $a_{ij} = \exp\left(-\frac{d_{ij}^2}{\sum_{i,j} d_{ij}^2}\right)$, where d_{ij} is the geodesic distance between the ith and jth cities. We consider the target to be the vector of temperature measurements over all 45 cities for a given day and the corresponding measurements from the previous day as the input $\mathbf{x}$. We construct noisy training targets by adding white Gaussian noise at a signal-to-noise (SNR) level of 5dB. The NMSE as a function of N for the sigmoid activation function is plotted in Figure 2. We observe that the ELMG outperforms ELM by a signficant margin for all K, particularly at small sample sizes. As N is increased, the performance of ELM and ELMG almost coincide. The NMSE obtained for different N and K values for all the three activation functions is listed in Table I.

B. fMRI data for cerebellum

We next consider the functional magnetic resonance imaging (fMRI) data obtained for the cerebellum region of the brain used in [24] .The data is available publicly at https://openfmri.org/dataset/ds000102. The data consists of the intensity values at 4000 different voxels obtained from the fMRI of the cerebellum region. The graph is obtained by mapping the cerebellum voxels anatomically following the atlas template [25], [26]. We refer to [24] for details of graph construction and associated signal extraction. We consider

ISBN 978-90-827970-1-5 © EURASIP 2018

the first 1000 nodes in our analysis. Our goal is to use the intensity values at the first 100 vertices as input $\mathbf{x} \in \mathbb{R}^{100}$ and make predictions for the remaining 900 vertices, which forms the output $\mathbf{t} \in \mathbb{R}^{900}$. We have a total of 295 graph signals corresponding to different measurements from a single subject. We use a one half of the data for training and the other half for testing. As with the earlier experiment, we construct noisy training targets at a signal-to-noise (SNR) level of 5dB. The NMSE of the prediction mean for testing data, averaged over 100 different random partitions of the dataset and different realizations of $\mathbf{a}, \mathbf{b}$ and noise, for the three activation functions is listed in Table II. As also seen from Figure 3, the trend remains the same as like the case of temperature data: ELMG outperforms ELM when the number of training samples is small, irrespective of the activation function and the number of neurons K.

VI. Conclusions

We bring together extreme learning machines and graph signal processing. Using the assumption that the signal to be predicted is smooth over a graph, the relevant regression problem uses graph-Laplacian matrix as well as a kernel between the observed inputs. The resulting solution has an interpretation of providing simultaneous smoothness across training samples and across graph nodes. Our hypothesis – the graph knowledge will improve performance of extreme learning machine – is verified by experiments on real data.

References

[1] G.-B. Huang, H. Zhou, X. Ding, and R. Zhang, "Extreme learning machine for regression and multiclass classification," *J. Trans. Sys. Man Cyber. Part B*, vol. 42, no. 2, pp. 513–529, Apr. 2012.

[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, Artificial Brains.

[3] G. Huang, G.-B. Huang, S. Song, and K. You, "Trends in extreme learning machines: A review," *Neural Networks*, vol. 61, no. Supplement C, pp. 32 – 48, 2015.

[4] A.A. Mohammed, R. Minhas, Q.M. Jonathan Wu, and M.A. Sid-Ahmed, "Human face recognition based on multidimensional pca and extreme learning machine," *Pattern Recognition*, vol. 44, no. 10, pp. 2588 – 2597, 2011, Semi-Supervised Learning for Visual Content Analysis and Understanding.

[5] T. Hussain, S. M. Siniscalchi, C. C. Lee, S. S. Wang, Y. Tsao, and W. H. Liao, "Experimental study on extreme learning machine applications for speech enhancement," *IEEE Access*, vol. PP, no. 99, pp. 1–1, 2017.

[6] G.-B. Huang, L. Chen, and C.-K. Siew, "Universal approximation using incremental constructive feedforward networks with random hidden nodes," *Trans. Neur. Netw.*, vol. 17, no. 4, pp. 879–892, July 2006.

[7] M. Luo, L. Zhang, J. Liu, J. Guo, and Q. Zheng, "Distributed extreme learning machine with alternating direction method of multiplier," *Neurocomputing*, vol. 261, no. Supplement C, pp. 164 – 170, 2017, Advances in Extreme Learning Machines (ELM 2015).

[8] J. Tang, C. Deng, and G. B. Huang, "Extreme learning machine for multilayer perceptron," *IEEE Trans. Neural Networks Learning Syst.*, vol. 27, no. 4, pp. 809–821, April 2016.

[9] B. Jin, Z. Jing, and H. Zhao, "Incremental and decremental extreme learning machine based on generalized inverse," *IEEE Access*, vol. 5, pp. 20852–20865, 2017.

[10] 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, 2013.

[11] 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, no. 5, pp. 80–90, 2014.

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

[13] R. I. Kondor and J. Lafferty, "Diffusion kernels on graphs and other discrete structures," *Proc. ICML*, pp. 315–322, 2002.

[14] Corinna Cortes and Mehryar Mohri, "On transductive regression," in *Proc. 19th Int. Conf. Neural Info. Process. Syst.*, Cambridge, MA, USA, 2006, pp. 305–312, MIT Press.

[15] 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*, pp. 173 – 198. Butterworth-Heinemann, 2018.

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

[17] F. R. K. Chung, *Spectral Graph Theory*, AMS, 1996.

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

[19] A. Sandryhaila and J. M. F. Moura, "Discrete signal processing on graphs: Frequency analysis," *IEEE Trans. Signal Process.*, vol. 62, no. 12, pp. 3042–3054, 2014.

[20] S. Chen, A. Sandryhaila, J. M. F. Moura, and J. Kovacevic, "Signal recovery on graphs: Variation minimization," *IEEE Trans. Signal Process.*, vol. 63, no. 17, pp. 4609–4624, Sept 2015.

[21] S. Chen, R. Varma, A. Sandryhaila, and J. Kovacevic, "Discrete signal processing on graphs: Sampling theory," *IEEE Transactions on Signal Processing*, vol. 63, no. 24, pp. 6510–6523, Dec 2015.

[22] R. A. Horn and C. R. Johnson, Eds., *Matrix Analysis*, Cambridge University Press, New York, NY, USA, 1986.

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

[24] H. Behjat, U. Richter, D. Van De Ville, and L. Sörnmo, "Signal-adapted tight frames on graphs," *IEEE Trans. Signal Process.*, vol. 64, no. 22, pp. 6017–6029, Nov 2016.

[25] H. Behjat, N. Leonardi, L. Sörnmo a, and D. Van De Ville, "Anatomically-adapted graph wavelets for improved group-level fmri activation mapping," *NeuroImage*, vol. 123, pp. 185 – 199, 2015.

[26] J. Diedrichsen, Joshua H. Balsters, Jonathan Flavell, Emma Cussans, and Narender Ramnani, "A probabilistic mr atlas of the human cerebellum," *NeuroImage*, vol. 46, no. 1, pp. 39 – 46, 2009.

ISBN 978-90-827970-1-5 © EURASIP 2018