Serdica Journal of Computing 18(1), 2024, pp. 1-25, 10.55630/sjc.2024.18.1-25
Published by Institute of Mathematics and Informatics, Bulgarian Academy of Sciences

Integrating Graph Structures into Kernel Regression Models

Eddie Pei[1], Ernest Fokoué[2]

[1]Munsell Color Science Laboratory,
Rochester Institute of Technology, New York, United States
ep2667@rit.edu

[2]School of Mathematical Sciences,
Rochester Institute of Technology, New York, United States
epfeqa@rit.edu

Abstract

Kernel methods are highly flexible and powerful tools for capturing complex, nonlinear relationships in data. In this paper, we propose a substantial extension of existing network-based regression models by integrating kernel methods with graph-theoretic constraints. Our approach builds upon the foundational work of Li et al. [1], who incorporated network cohesion into generalized linear models (GLMs). We extend their framework by introducing a kernelized regression model that allows for the modeling of nonlinear interactions while leveraging network data. This kernelized framework relaxes the linearity constraints of GLMs, making the model more versatile and capable of capturing complex patterns in high-dimensional spaces.

We demonstrate the effectiveness of our approach using simulated and real-world datasets, such as the Teenage Friends and Lifestyle Study. Results show that our kernelized network regression model not only outperforms traditional linear and generalized linear models in predictive accuracy but also scales efficiently to larger datasets. Our work represents a significant advancement in the modeling of network-linked data, providing a robust, scalable, and interpretable framework that extends the application of kernel methods beyond their traditional constraints.

Future directions include exploring more complex graph structures, such as weighted and directed graphs, and developing optimized algorithms for even larger datasets.

Received: June 13, 2024, *Accepted:* October 15, 2024, *Published:* October 25, 2024
Citation: Eddie Pei, Ernest Fokoué, Integrating Graph Structures into Kernel Regression Models, Serdica Journal of Computing 18(1), 2024, pp. 1-25, https://doi.org/10.55630/sjc.2024.18.1-25

Keywords: Regression Analysis, Kernel Regression, Regularization, Network Data, Graph Laplacian, Network Cohesion

ACM Computing Classification System 2012: Computing methodologies → Machine learning → Machine learning algorithms

1 Introduction

Network data, which captures relationships between entities, has become prevalent in various fields such as social media analysis, biology, economics, and sociology. The challenge of incorporating network structures into predictive modeling has led to significant advances in statistical machine learning. Traditionally, linear regression models have served as the cornerstone of supervised learning, and their adaptation to network data has been a key research area.

One of the most notable contributions in this domain is the work by Li et al. [1], which introduced the idea of incorporating network cohesion into regression models, allowing for observation-dependent intercepts based on network structures. Li et al.'s approach, which extended the classical linear regression model to accommodate generalized linear models (GLMs), opened new possibilities for network-based prediction by leveraging graph-based structures. In this framework, the relationship between entities in the network is represented through a graph, and these relationships are incorporated into the model via an adjacency matrix.

The model proposed by Li et al. demonstrated that network cohesion, when modeled effectively, could significantly enhance predictive performance. However, their framework is constrained by the inherent linearity of GLMs, limiting its capacity to capture nonlinear dependencies in the data. It's interesting to note that aside from [1], the literature on theory, methodology, computation, and applications of graph theoretic methods now abounds [2–6].

In this work, we propose a substantial extension of Li et al.'s model by introducing a kernelized regression framework that allows for more flexible and powerful modeling of nonlinear relationships within the network structure. Kernel methods have become a cornerstone of modern machine learning due to their ability to implicitly map data into high-dimensional feature spaces, capturing complex interactions that would otherwise be missed by linear models. By combining kernel regression with graph-based network data, we enable the model to handle nonlinearities while preserving the structure of relationships captured by the graph. This kernelized approach retains the interpretability and flexibility of network models while vastly improving their capacity for handling nonlinearity in high-dimensional settings.

One of the central innovations of this work lies in the incorporation of the adjacency matrix into the kernel function, allowing us to capture both the intrinsic properties of the data and the relationships between them. We work with undirected and unweighted graphs, where the incidence matrix is binary, i.e., $a_{ij} = 1$ if node i is connected to node j, and $a_{ij} = 0$ otherwise. This simple yet effective representation of relationships suffices for our purpose of modeling social networks among youngsters, as seen in the Teenage Friends and Lifestyle Study dataset [7].

We build on Li et al.'s work by replacing their linear modeling approach with a kernel-based regression model that uses a regularized kernel learning machine. In particular, we extend the model to account for nonlinearity in both the input features and the relationships encoded by the network structure. Through this enhancement, we demonstrate that kernelizing the regression model with graph constraints leads to better performance in both interpretability and predictive accuracy.

In our experiments on real-world social network data, we observe significant improvements over standard GLM-based models, with the kernelized version capturing subtle nonlinear patterns that the earlier models missed. Li et al.'s work rightly deserves to be deemed seminal in this context and provides a detailed account of linear regression with network cohesion, specifically deriving both the algorithmic details of a sound theory and a promising application [1]. They even provide a package in the R software environment to help with the exploration of a variety of network data-related statistical machine learning activities [8].

Building from their work, which established the linear regression model and later expanded to generalized linear models for broader application, this research departs from the interpretability that motivated Li to adopt the GLM family. It instead emphasizes the versatility, flexibility, and predictive capacity of the function space, selecting kernel regression as the central approach. In a spirit similar to Li's research, we build a regularized kernel regression framework with a regularizing component induced by graph theoretic constraints assumed to underlie the accompanying network data.

This research's quintessential contribution lies in improving the predictive performance of kernel regression by appropriately induced graph-theoretic constraints similar to those used by Li et al. [1] on generalized linear models.

Throughout this research, we focus on the scenario where we are given a data set

$$\mathscr{D}_n = \{(\mathbf{x}_j, y_j) \overset{iid}{\sim} p_{xy}(\mathbf{x}, y),\ \mathbf{x}_j \in \mathscr{X},\ y_j \in \mathbb{R},\ j = 1, \ldots, n\}.$$

We assume a regression learning model under homoscedastic Gaussian noise, specifically, $(Y_i|f(\cdot), \mathbf{x}_i, \sigma^2) \overset{iid}{\sim} N(f(\mathbf{x}_i), \sigma^2)$, where $f \in \mathscr{H}_{\mathcal{K}}$ with the corresponding function space is defined by

$$\mathscr{H}_{\mathcal{K}} := \left\{ \mathbf{x} \mapsto \sum_{j=1}^{n} \mathrm{w}_j \mathcal{K}(\mathbf{x}, \mathbf{x}_j) + \alpha(\mathbf{x}),\ \alpha(\mathbf{x}) \in \mathbb{R},\, \mathrm{w}_j \in \mathbb{R},\, j \in [n] \right\}, \tag{1}$$

where $\alpha(\mathbf{x})$ represents the network contribution of $\mathbf{x}$.

We will see in subsequent sections that α_i during training is replaced by $\alpha_{\texttt{new}}$ for predicting out-of-training samples, specifically for the $\mathbf{x}_{\texttt{new}}$ whose response value is to be predicted for members of the test set. In this context, the kernel $\mathcal{K}$ is a bivariate function defined on the input space. For our purposes, the kernel measures the similarity between any two given elements from $\mathscr{X}$, and is therefore defined as:

$$\begin{aligned} \mathcal{K}: \quad & \mathscr{X} \times \mathscr{X} \longrightarrow \mathbb{R}_+ \\ & (\mathbf{x}_l, \mathbf{x}_m) \longmapsto \mathcal{K}(\mathbf{x}_l, \mathbf{x}_m). \end{aligned} \tag{2}$$

For clarity, one of the most commonly used kernels in practical applications, is the so-called Gaussian Radial Basis Function (RBF) kernel, which is very flexible and versatile and defined as follows:

$$\mathcal{K}(\mathbf{x}_l, \mathbf{x}_m) = \exp\Big(-\gamma \|\mathbf{x}_l - \mathbf{x}_m\|_2^2 \Big). \tag{3}$$

There are very many other kernels used by practitioners and methodologists for various applications as we will see later in our computational demonstrations. As we will see in the rest of this paper, once a kernel $\mathcal{K}$ is selected for the task of interest, the resulting Gram matrix $\boldsymbol{K} = (\mathcal{K}(\mathbf{x}_l, \mathbf{x}_m)), l.m = 1, \ldots, n$ is formed and, became a central element in subsequent analyses. It is readily apparent that the Gram matrix $\boldsymbol{K}$ fulfills a role analogous to that of the design matrix or data matrix $\boldsymbol{X}$ in linear models:

$$\boldsymbol{K} := \begin{bmatrix} \mathcal{K}(\mathbf{x}_1, \mathbf{x}_1) & \mathcal{K}(\mathbf{x}_1, \mathbf{x}_2) & \cdots & \mathcal{K}(\mathbf{x}_1, \mathbf{x}_n) \\ \mathcal{K}(\mathbf{x}_2, \mathbf{x}_1) & \mathcal{K}(\mathbf{x}_2, \mathbf{x}_2) & \cdots & \mathcal{K}(\mathbf{x}_2, \mathbf{x}_n) \\ \vdots & \vdots & \cdots & \vdots \\ \mathcal{K}(\mathbf{x}_i, \mathbf{x}_1) & \mathcal{K}(\mathbf{x}_i, \mathbf{x}_2) & \cdots & \mathcal{K}(\mathbf{x}_i, \mathbf{x}_n) \\ \vdots & \vdots & \cdots & \vdots \\ \mathcal{K}(\mathbf{x}_n, \mathbf{x}_1) & \mathcal{K}(\mathbf{x}_n, \mathbf{x}_2) & \cdots & \mathcal{K}(\mathbf{x}_n, \mathbf{x}_n) \end{bmatrix}. \tag{4}$$

Some learning machines, including support vector machines, require the kernel to be positive definite in order to guarantee the convergence and stability properties of the learning process. Our choice of the function space in Equation (1) aims to not only extend but also enhance the methodologies proposed by Li et al. [1] in terms of both predictive accuracy and modeling flexibility.

Kernel regression models provide significant versatility since they can handle input spaces that are not necessarily real-valued, thus allowing for a more adaptable modeling framework. Moreover, the added benefit is the ability of kernel regression learning machines to represent arbitrarily complex underlying regression functions.

In fact, kernel regression and kernel methods have grown tremendously in popularity, and they are continually used, further developed, and improved all the time. Kernel methods have never ceased to gain more popularity in modeling, and many software packages are continually built around them [6, 9].

Gaussian Processes, for instance, have provided one of the most commonly used kernel methods and become fashionable in machine learning, having proven to be formidable learning machines with excellent predictive performances on high dimensional [10–15]. The phenomenal success of support vector classifiers was immediately followed by the arrival on the scene of support vector regression, with the seminal paper featuring support vector regression appearing in [16] and [17].

The relevance vector machine [18] is another example of a kernel learning machine that yielded excellent predictive performance in high dimensional regression. Central to all the kernel learning machines is the need for suitable regularization owing to their inherent ill-posedness. It is found that in their practical form used for actually building the kernel regression machine, the learning task is done via the optimization of the following penalized (regularized) empirical risk functional:

$$\widehat{R}_{\lambda,n}(f) = \frac{1}{2n}\sum_{i=1}^{n}(y_i - f(\mathbf{x}_i))^2 + \frac{\lambda}{2}\|f\|_{\mathcal{K}}^2 \tag{5}$$

where $\|f\|_{\mathcal{K}}^2$ plays an extremely important role in addressing the typically ill-posed learning problem inherent in kernel learning machines. Equation (5) is clearly very general and indeed generic, yet it provides the perfect vehicle for us to formulate the two-fold contribution of the present manuscript as we will see in the subsequent sections.

2 Kernel regression with double penalization

Once again, the chief aim of this manuscript is to extend [1] using both kernelization to capture subtle nonlinearities in the patterns of functional relationships, and the network relationships among the sampling units to further sharpen our predictions. To this end, our two-fold contribution not only suitably penalized the weights in the kernel expansion, but also integrates the graph Laplacian induced penalization that captures the patterns of relationship among the sampling units. It is fair to claim that our treatment is sufficiently detailed and thorough.

2.1 First regularization via penalization on weights

It is noteworthy that the function space described in Equation (1) is a generalization of a scenario in which $\alpha(\mathbf{x}) = \alpha$ is a constant. In such cases of constant α, the representer theorem of [19, 20] results in an empirical risk function that is expressed in terms of the weights, namely

$$\widehat{R}_{\lambda,n}(\boldsymbol{w}) = \frac{1}{2n}(\boldsymbol{Y} - \alpha\mathbb{1}_n - \boldsymbol{K}\boldsymbol{w})^\top(\boldsymbol{Y} - \alpha\mathbb{1}_n - \boldsymbol{K}\boldsymbol{w}) + \frac{\lambda}{2}\boldsymbol{w}^\top\boldsymbol{K}\boldsymbol{w} \tag{6}$$

which yields the nice and desirable closed-form expression

$$\widehat{\boldsymbol{w}} = (\boldsymbol{K} + n\lambda\boldsymbol{I}_n)^{-1}\boldsymbol{Y} \qquad \text{and} \qquad \widehat{\alpha} = \bar{\boldsymbol{Y}} - n^{-1}\mathbf{1}_n\boldsymbol{K}\widehat{\boldsymbol{w}}. \tag{7}$$

One of the distinct advantages of Equation (7) is that it provides estimates in closed form, which can be readily obtained computationally.

2.2 Regularized kernel regression via network cohesion

Upon encountering this regularization framework, a natural question arises about how to integrate or adapt the network data into it. As remarked by Li et al., the model inherent in Equation (6) uses an intercept α that remains the same for all the observations in the training set $\mathscr{D}_n$. Providentially, it turns out that making the intercept observation-dependent naturally provides a mechanism for incorporating the network data into the model via the graph theoretical derivations. This method of incorporation is also discussed in other works [1,3–5,21–23]. Specifically, [1] developed multiple linear regression incorporating network cohesion, formulated as:

$$Y_i = \alpha_i + \boldsymbol{\beta}^\top\mathbf{x}_i + \varepsilon_i, \quad \text{with} \quad \mathbf{x}_i \in \mathbb{R}^p, \boldsymbol{\beta} \in \mathbb{R}^p \quad \text{and} \quad \varepsilon_i \overset{iid}{\sim} N(0, \sigma^2), \tag{8}$$

which we are extending and adapting within the context of kernel learning in Equation (9) below, namely $Y_i = f(\mathbf{x}_i) + \varepsilon_i$, $i \in [n]$, with

$$f(\mathbf{x}_i) = \alpha_i + \sum_{j=1}^{n} \mathrm{w}_j\mathcal{K}(\mathbf{x}_i, \mathbf{x}_j) \quad \text{and} \quad \varepsilon_i \overset{iid}{\sim} N(0, \sigma^2), \quad i \in [n]. \tag{9}$$

Crucially, the α_i values are incorporated in as contributors to the model from the network data, as detailed in Equation (1). We, therefore, end up having $2n$ parameters to estimate: n from the vector $\boldsymbol{\alpha}$ and n from the vector $\boldsymbol{w}$. In practice, with only n observations in the data, estimating $2n$ quantities is inherently ill-posed. To address this challenge, regularization ends up being the approach used to circumvent this difficulty, with assumptions like $A_{uv} = 1$ and $A_{uu} = 0$ in the adjacency matrix following the approach outlined by Li et al. Specifically, the resulting regularized kernel regression model, in its raw form, is given by:

$$\widehat{R}_{\lambda,n}(\boldsymbol{\alpha}, \boldsymbol{w}) = (\boldsymbol{Y} - \boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{w})^\top(\boldsymbol{Y} - \boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{w}) + \lambda\boldsymbol{\alpha}^\top\boldsymbol{L}\boldsymbol{\alpha}. \tag{10}$$

The vector n-dimensional vector $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, ..., \alpha_n) \in \mathbb{R}^n$ is the vector of the individual effects of every node, the vector $\boldsymbol{w} = (\mathrm{w}_1, \mathrm{w}_2, ..., \mathrm{w}_n)^\top \in \mathbb{R}^n$ is the vector of regression coefficients, and

$$\boldsymbol{\alpha}^\top\boldsymbol{L}\boldsymbol{\alpha} = \sum_{(u,v)\in\boldsymbol{E}} (\alpha_u - \alpha_v)^2. \tag{11}$$

In the model, it is assumed that the graph $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ captures the links between observations derived from network data, where $\boldsymbol{V} = \{1, 2, ..., n\}$ is the node set of the graph, and $\boldsymbol{E} \subset \boldsymbol{V} \times \boldsymbol{V}$ is the edge set. The corresponding adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ is the graph Laplacian $\mathcal{G} = (\boldsymbol{V}, \boldsymbol{E})$ which is given by $\boldsymbol{L} = \boldsymbol{D} - \boldsymbol{A}$, where $\boldsymbol{D} = diag(d_1, d_2, ..., d_n)$ is the degree matrix, with each node degree d_v calculated as $d_v = \sum_{u\in\boldsymbol{E}} A_{uv}$, following the methodology outlined in [24].

Through straightforward computations, including matrix differentiation as detailed in [25], it is possible to derive estimates of both $\boldsymbol{\alpha}$ and $\boldsymbol{w}$. The estimators are computed as follows:

$$(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{w}}) = (\tilde{\boldsymbol{K}}^\top\tilde{\boldsymbol{K}} + \lambda\boldsymbol{M})^{-1}\tilde{\boldsymbol{K}}^\top\boldsymbol{Y}, \tag{12}$$

where $\boldsymbol{Y} = (y_1, \ldots, y_n)$ is the n-dimensional vector of response values,

$$\tilde{\boldsymbol{K}} = [I_{n\times n}, \boldsymbol{K}] \quad \text{and} \quad \boldsymbol{M} = \begin{bmatrix} L & 0_{n\times n} \\ 0_{n\times n} & 0_{n\times n} \end{bmatrix}.$$

It bears emphasizing that the straightforward derivation of the key components of this regression framework is one of its key appeals. This appeal further carries through when we combine kernel expansion weights regularization with network cohesion regularization.

2.3 Kernel regression with weights and network regularization

While the above straightforward adaptation of Li's research to kernel regression provides several advantages, we considered it advantageous to combine traditional regularization on $\boldsymbol{w}$ with the $\boldsymbol{\alpha}$ derived from the network. As a result, a more complete and certainly more powerful framework is derived, as seen in the following regularized objective function depicted in Equation (13):

$$\widehat{R}_{\lambda,n}(\boldsymbol{\alpha}, \boldsymbol{w}) = (\boldsymbol{Y} - \boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{w})^\top(\boldsymbol{Y} - \boldsymbol{\alpha} - \boldsymbol{K}\boldsymbol{w}) + \lambda\boldsymbol{\alpha}^\top\boldsymbol{L}\boldsymbol{\alpha} + \psi\boldsymbol{w}^\top\boldsymbol{K}\boldsymbol{w}. \tag{13}$$

Theorem 1. *Based on Equation (13), the derivation of the corresponding estimates for $\boldsymbol{\alpha}$ and $\boldsymbol{w}$ is straightforward and in the following form.*

$$(\widehat{\boldsymbol{\alpha}}, \widehat{\boldsymbol{w}}) = (\tilde{\boldsymbol{K}}^\top\tilde{\boldsymbol{K}} + \psi\boldsymbol{N} + \lambda\boldsymbol{M})^{-1}\tilde{\boldsymbol{K}}^\top\boldsymbol{Y}, \tag{14}$$

where $\tilde{\boldsymbol{K}} = [I_{n\times n}, \boldsymbol{K}]$, $\boldsymbol{N} = \begin{bmatrix} 0_{n\times n} & 0_{n\times n} \\ 0_{n\times n} & I_{n\times n} \end{bmatrix}$ *and* $\boldsymbol{M} = \begin{bmatrix} L & 0_{n\times n} \\ 0_{n\times n} & 0_{n\times n} \end{bmatrix}$.

Proof. By using standard matrix-vector derivation, we get the following:

$$\frac{dL(\boldsymbol{\alpha}, \boldsymbol{w})}{d\boldsymbol{\alpha}} = -2\boldsymbol{K}^\top(\boldsymbol{Y} - \boldsymbol{K}\boldsymbol{w} - \boldsymbol{\alpha}) + 2\psi\boldsymbol{I}_{n\times n}\boldsymbol{w} = 0$$
$$\frac{dL(\boldsymbol{\alpha}, \boldsymbol{w})}{d\boldsymbol{w}} = -2\boldsymbol{I}_{n\times n}\boldsymbol{K}^\top(\boldsymbol{Y} - \boldsymbol{K}\boldsymbol{w} - \boldsymbol{\alpha}) + 2\lambda\boldsymbol{L}\boldsymbol{\alpha} = 0$$

By solving the above equations, it is straightforward to get

$$\begin{aligned} \boldsymbol{K}^\top\boldsymbol{Y} - \boldsymbol{K}^\top\boldsymbol{K}\boldsymbol{w} - \boldsymbol{K}^\top\boldsymbol{\alpha} - \psi\boldsymbol{I}\boldsymbol{w} &= 0 \\ \boldsymbol{Y} - \boldsymbol{K}\boldsymbol{w} - \boldsymbol{\alpha} - \lambda\boldsymbol{L}\boldsymbol{\alpha} &= 0 \end{aligned}$$

which can be written as:

$$\begin{aligned} \boldsymbol{K}^\top\boldsymbol{Y} &= (\boldsymbol{K}^\top\boldsymbol{K} + \psi\boldsymbol{I})\boldsymbol{w} + \boldsymbol{K}^\top\boldsymbol{\alpha} \\ \boldsymbol{Y} &= \boldsymbol{K}\boldsymbol{w} + (\boldsymbol{I} + \lambda\boldsymbol{L})\boldsymbol{\alpha} \end{aligned}$$

The result could be written in matrix form:

$$\begin{pmatrix} \boldsymbol{I}_n \\ \boldsymbol{K}^\top \end{pmatrix} \boldsymbol{Y} = \begin{pmatrix} \boldsymbol{I}_n + \lambda\boldsymbol{L} & \boldsymbol{K} \\ \boldsymbol{K}^\top & \boldsymbol{K}^\top\boldsymbol{K} + \psi\boldsymbol{I}_n \end{pmatrix} \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{w} \end{pmatrix},$$

which equals:

$$\begin{pmatrix} I_n \\ \boldsymbol{K}^\top \end{pmatrix} \boldsymbol{Y} = \left[\begin{pmatrix} \boldsymbol{I}_n & \boldsymbol{K} \\ \boldsymbol{K} & \boldsymbol{K}^\top\boldsymbol{K} \end{pmatrix} + \lambda \begin{pmatrix} \boldsymbol{L} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{0} \end{pmatrix} + \psi \begin{pmatrix} \boldsymbol{0} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{I}_n \end{pmatrix} \right] \begin{pmatrix} \boldsymbol{\alpha} \\ \boldsymbol{w} \end{pmatrix}.$$

□

It is remarkable how straightforward the adaptation of network cohesion to kernel regression turns out to be, especially considering how substantial the gain in predictive performance ends up being in practice, as seen later in the computational demonstrations. As mentioned previously, one of the principal advantages of kernel methods is their ability to handle non-numeric input spaces, provided that suitable kernels (measures of similarity) are well-defined.

2.4 Obtaining predictions in training and testing

Since improvement in predictive performance is one of the main motivations for resorting to kernel methods, we now show the straightforward nature of the prediction function in this case. Having obtained $\widehat{\boldsymbol{\alpha}}$ and $\widehat{\boldsymbol{w}}$, it is straightforward to see that for all $(\mathbf{x}_i, y_i) \in \mathscr{D}_n$

$$\widehat{f}(\mathbf{x}_i) = \widehat{\alpha}_i + \sum_{j=1}^{n} \widehat{\mathrm{w}}_j \mathcal{K}(\mathbf{x}_i, \mathbf{x}_j), \tag{15}$$

so that for the whole training set $\mathscr{D}_n$, the n-dimensional vector $\widehat{\boldsymbol{Y}}$ of in-sample predicted (fitted) values is given by

$$\widehat{\boldsymbol{Y}} = \tilde{\boldsymbol{K}}(\tilde{\boldsymbol{K}}^\top \tilde{\boldsymbol{K}} + \psi \boldsymbol{N} + \lambda \boldsymbol{M})^{-1} \tilde{\boldsymbol{K}} \boldsymbol{Y} = \boldsymbol{H}(\lambda, \psi) \boldsymbol{Y}, \tag{16}$$

where

$$\boldsymbol{H}(\lambda, \psi) = \tilde{\boldsymbol{K}}(\tilde{\boldsymbol{K}}^\top \tilde{\boldsymbol{K}} + \psi \boldsymbol{N} + \lambda \boldsymbol{M})^{-1} \tilde{\boldsymbol{K}} \tag{17}$$

is the so-called "hat" matrix. Given that one of the principal motivations behind adopting kernel regression is the construction of optimal predictive regression models, it is important to focus on the out-of-sample prediction mechanism. More specifically, we seek to make $\mathbb{E}[\widehat{f}(\mathbf{x}_{\texttt{new}})]$ small, where

$$\widehat{f}(\mathbf{x}_{\texttt{new}}) = \widehat{\alpha}_{\texttt{new}} + \sum_{j=1}^{n} \widehat{\mathrm{w}}_j \boldsymbol{K}(\mathbf{x}_{\texttt{new}}, \mathbf{x}_j). \tag{18}$$

With n training observations and m testing observations, the resulting Laplacian is $(n+m) \times (n+m)$ matrix that admits the decomposition of the form:

$$\tilde{\boldsymbol{L}} = \begin{pmatrix} \boldsymbol{L}_{ss} & \boldsymbol{L}_{st} \\ \boldsymbol{L}_{ts} & \boldsymbol{L}_{tt} \end{pmatrix}, \tag{19}$$

where $\boldsymbol{L}_{ss}$ corresponds to the n training observations, $\boldsymbol{L}_{tt}$ to the m observations from the test set, while $\boldsymbol{L}_{st}$ and $\boldsymbol{L}_{ts}$ correspond to the relationships between the members of the training set and those of the test set. The complete $(n+m)$-dimensional individual effect vector is $\boldsymbol{\alpha}_{\texttt{all}} = (\boldsymbol{\alpha}_s^\top, \boldsymbol{\alpha}_t^\top)^\top$, where $\boldsymbol{\alpha}_s \in \mathbb{R}^n$ refers to the training data, and $\boldsymbol{\alpha}_t \in \mathbb{R}^m$ refers to the test data for which predictions outside of training are sought. The following step is to set the optimization problem:

$$\widehat{\boldsymbol{\alpha}}_t = \underset{\boldsymbol{\alpha}_t}{\texttt{argmin}}\Big\{(\widehat{\boldsymbol{\alpha}_s}^\top, \boldsymbol{\alpha}_t^\top)^\top \tilde{\boldsymbol{L}}(\widehat{\boldsymbol{\alpha}}_s^\top, \boldsymbol{\alpha}_t^\top)\Big\}, \tag{20}$$

which in turn yields the following desired result:

$$\widehat{\boldsymbol{\alpha}}_t = -\boldsymbol{L}_{tt}^{-1}\boldsymbol{L}_{ts}\widehat{\boldsymbol{\alpha}}_s. \tag{21}$$

Given the estimate $\widehat{\boldsymbol{\alpha}}_t$, it is easy to make out of sample predictions. Note that $\widehat{\alpha}_{\texttt{new}}$ in Equation (18) is simply the same of as $\widehat{\boldsymbol{\alpha}}_t$ for a single prediction.

3 Computational explorations and demonstrations

We herein use both simulated and real-world data to assess the performance of our proposed models and learning methods. We specifically compare our proposed kernel regression with network cohesion against a wide variety of other scenarios. We compare our work to [1] featuring the generalized linear model with the network cohesion model. We also compare our work against the generic multiple linear regression model without network information with the finality of checking evidence for the effect of the non-linearity of kernels and the effect of network data on the overall performance of regression estimators. Importantly, we make comparisons against standard kernel regression paradigms like support vector regression, relevance vector regression, and Gaussian process regression to further assess and measure network data's effect on the predictive performance.

In the spirit of predictive optimality, we focus on estimates of the generalization error given by many random replications of the test error. We generate 50 replicates of the test error and perform our comparisons based on those. Throughout our simulations, we assume and generate data from a model of the form:

$$Y_i = \alpha_i + f(\mathbf{x}_i) + \varepsilon_i, \quad i = 1, \ldots, n, \tag{22}$$

and we further assume homoscedastic Gaussian noise, so that $\varepsilon_i \overset{iid}{\sim} N(0, \sigma^2)$, for some noise variance σ^2 suitably chosen in keeping with signal to noise ratio. For clarity, we specify the target signal-to-noise ratio for each simulated example. Regarding the values of the α_i's, we either use the degree of the vertex or the same scheme used by Li's work. Regarding the kernels, we simply resort to the

Algorithm 1 Kernelized Regression with Network Constraints

```
Require: Data $D_n = \{(x_i, y_i)\}_{i=1}^n$ with features $x_i \in \mathbb{R}^p$, network adjacency
    matrix $A \in \mathbb{R}^{n\times n}$, kernel function $K$, regularization parameters $\lambda$ and $\psi$
Ensure: Predictions $\hat{y}$ for a test set $\{x_{\text{new}}\}$
 1: Construct the Kernel Matrix
 2: for each $i, j \in \{1, \ldots, n\}$ do
 3:     Compute kernel value $K(x_i, x_j)$
 4:     Store in Gram matrix $K_{ij} = K(x_i, x_j)$
 5: end for
 6: Regularize with Network Cohesion
 7: Form the Laplacian matrix $L = D - A$, where $D$ is the degree matrix and $A$
    is the adjacency matrix
 8: Construct the augmented kernel matrix $\tilde{K} = [I_n, K]$
 9: Construct the regularization matrix $M = \begin{bmatrix} L & 0 \\ 0 & 0 \end{bmatrix}$
10: Solve the Regularized Problem
11: Compute the weight vector $w$ and intercept vector $\alpha$ by solving:
        $$(\tilde{K}^T\tilde{K} + \lambda M + \psi I)\begin{bmatrix} \alpha \\ w \end{bmatrix} = \tilde{K}^T Y$$
    where $Y = (y_1, y_2, \ldots, y_n)^T$
12: Make Predictions
13: for each new data point $x_{\text{new}}$ do
14:     Compute the predicted value $\hat{y}(x_{\text{new}})$ as:
            $$\hat{y}(x_{\text{new}}) = \alpha_{\text{new}} + \sum_{j=1}^{n} w_j K(x_{\text{new}}, x_j)$$
15: end for
```

most commonly used ones, namely (a) the linear kernel, also referred to as the vanilla kernel, (b) the polynomial kernel, (c) the Gaussian Radial basis function kernel, (d) the Laplace kernel, and (e) the hyperbolic tangent kernel. Some other kernel functions can be found in [9, 26].

3.1 First simulated example

For our first simulation study, we consider the two-dimensional Euclidean space along with a nonlinear underlying regression function f defined in Equation (23). We also use a uniform network as shown in Figure (1). For the network contribution α, we simply use the degree of the network for each vertex. The variables are generated from a multivariate Gaussian distribution, and the error term ε is taken from a standard normal distribution. For the purposes of this computational demonstration, we repeatedly generate 200 data points, with 100 of those used for training and the remaining 100 used as test data for computing our test errors.

$$f(\mathbf{x}) = \mathrm{x}_1^2 + \mathrm{x}_2 \tag{23}$$

We conducted 50 replicates and compared the mean squared error (MSE). The results are displayed in Figure 2 and Table 1. As shown in Figure 2, the polynomial kernel model incorporating network cohesion outperforms all other models in this dataset. Similarly, it leads in terms of the lowest mean squared test error as recorded in Table 1. Generally, models that include network cohesion tend to surpass those that do not, demonstrating that the integration of network information significantly enhances the model's predictive performance.

We do 50 replicates and compare the mean squared error. The results are shown in Figure 2 and Table 1. In Figure 2, the polynomial kernel model incorporating network cohesion outperforms all other models in this dataset. Similarly, it leads in terms of the lowest mean squared test error as recorded in Table 1. Generally, models that include network cohesion tend to surpass those that do not, demonstrating that integrating network information significantly enhances the model's predictive performance.

3.2 Moderately challenging simulated data

We now consider an intriguing simulated example found in [27], herein referred to as Friedman1, with underlying function f given for each $\mathbf{x} = (\mathrm{x}_1, \mathrm{x}_2, \mathrm{x}_3, \mathrm{x}_4, \mathrm{x}_5)^\top \in \mathscr{X}$ by

$$f(\mathbf{x}) = 10\sin(\pi \mathrm{x}_1 \mathrm{x}_2) + 20\left(\mathrm{x}_3 - \frac{1}{2}\right)^2 + 10\mathrm{x}_4 + 5\mathrm{x}_5. \tag{24}$$

For this example, we use the bipartite network shown in Figure 3. For the data, we use the Friedman1 data set. We simulate 200 data points, 100 for training data, and 100 for test data. Since we only focus on the predicted performance, we eliminate the noise variables in the dataset. The error ϵ is from Normal distribution with $\boldsymbol{N}(0,2)$. The individual network parameter is given by:

$$\{\alpha_i = 2, \alpha_j = -2, where\ i = 1\cdots 100;\ j = 101\cdots 200\}.$$

Compared to the last experiment, the data structure is more complex. The response variable is shown in Equation (24).

We perform 50 replicates and compare the mean squared error. The results are shown in Figure 4 and Table 2. The outcomes are represented in Figure 4a, where models incorporating network cohesion consistently outperform others. Table 2 corroborates these findings, highlighting that the polynomial kernel regression with network cohesion is the most effective model for this dataset.

3.3 More complex simulated data

For the last simulation study, we use yet another dataset generated thanks to Friedman and his co-authors and found in [28] and herein referred to as the Friedman3 data. and the data structure is even more complex than the previous two data sets. We use the bipartite network in this simulation. Like the last two experiments, we simulate 200 data points, 100 for training, and 100 for test data. For the error, ϵ is from Normal distribution with $\boldsymbol{N}(0,0.1)$ and the network parameter α is given by

$$\{\alpha_i = 1,\ \alpha_j = -1, \quad where\ i = 1\cdots 100;\ j = 101\cdots 200\}.$$

The response variable is given in Equation (25).

$$f(\mathbf{x}) = \tan^{-1}\left(\frac{\mathrm{x}_2\mathrm{x}_3 - \frac{1}{\mathrm{x}_2\mathrm{x}_4}}{\mathrm{x}_1}\right). \tag{25}$$

In this case, the input space $\mathscr{X}$ is a subset of the four-dimensional Euclidean space $\mathbb{R}^4$, with each of the four variables measured on a different scale. Specifically, $0 \le \mathrm{x}_1 \le 100$, $20 \le (\mathrm{x}_2/2\pi) \le 280$, $0 \le \mathrm{x}_3 \le 1$ and $1 \le \mathrm{x}_4 \le 11$.

Similar to the last two experiments, We did 50 replicates and compared the mean squared error. The results are shown in Figure 5 and Table 3. According to Table 3, all the network cohesion models do extremely well. The polynomial kernel regression with the network cohesion model is the best one; furthermore, Table 5a shows all the network cohesion models have a smaller variance of test errors and smaller test errors than the others.

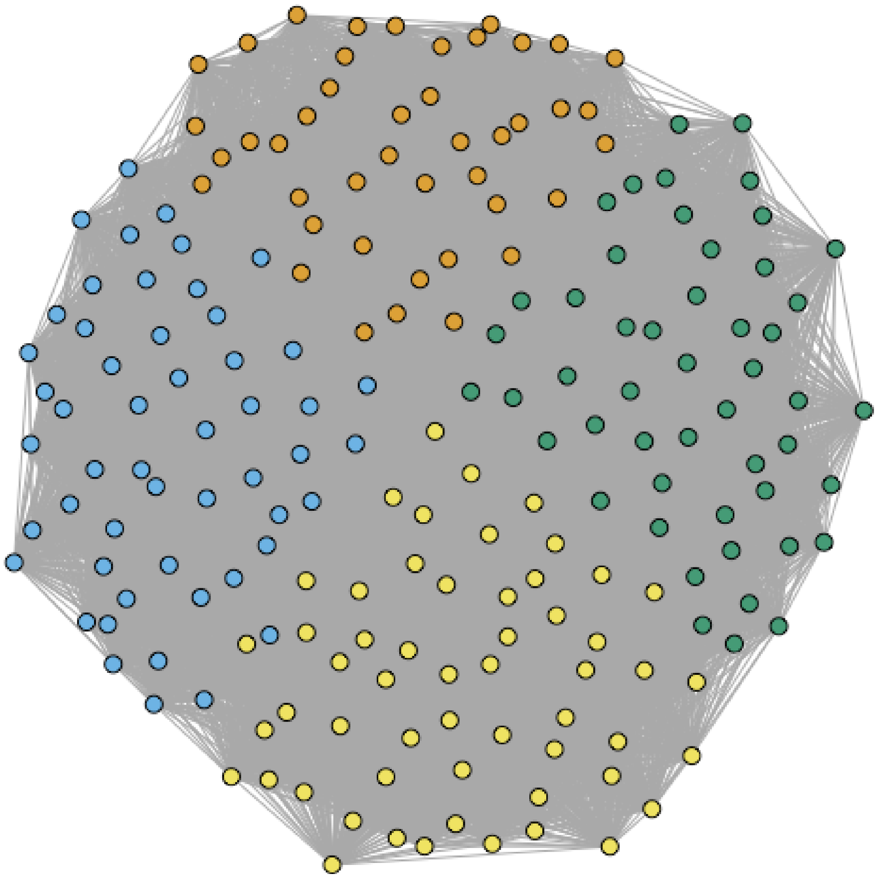
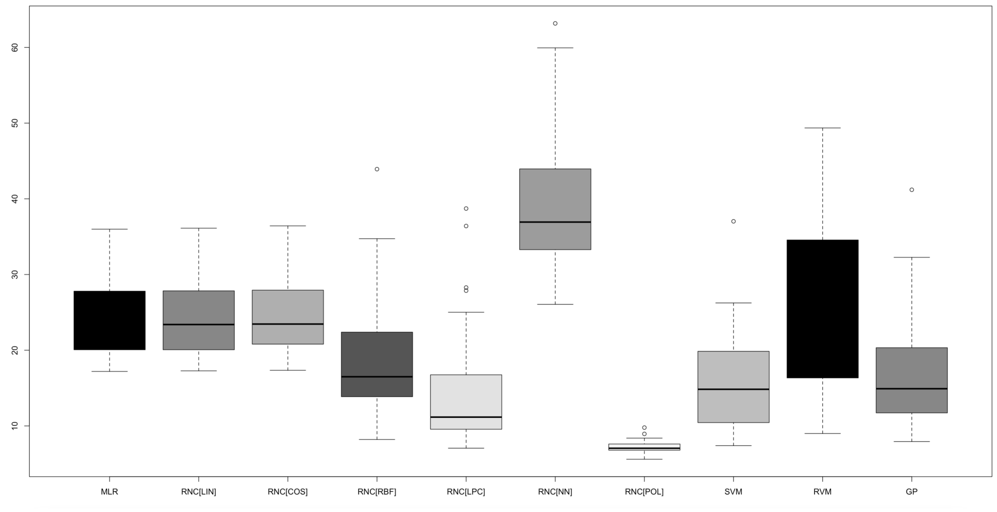


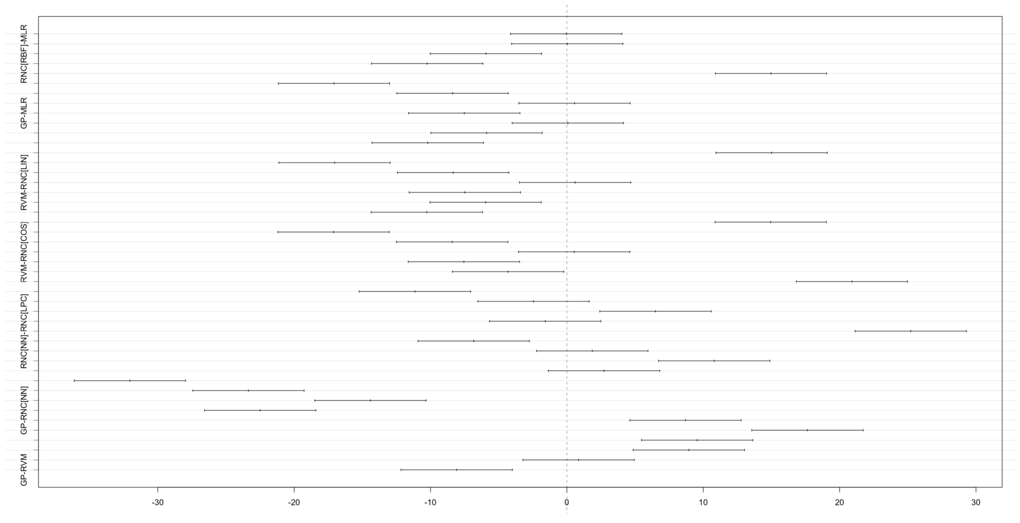
Figure 1: Uniform Networks, there are four groups in the network, which are shown as four colors.

Table 1: Mean squared training and test error with Uniform network. LIN: linear regression with network cohesion, COS: kernel regression with network cohesion models that use cosine kernel, RBF: Gaussian kernel, LPC: Laplace kernel, NN: hyperbolic tangent kernel, POL: polynomial kernel, MLR: multilinear regression, SVM: support vector machine, RVM: relevance support machine and GP: Gaussian processes for regression.

SN	MLR	LIN	COS	RBF	LPC	NN	POL	SVM	RVM	GP
Training Error	23.01	14.94	20.16	4.94	**4.59**	10.03	5.58	101.17	6.38	101.19
Test Error	24.26	24.22	24.29	18.33	14.01	39.23	**7.18**	15.88	24.82	16.74



(a) Uniform Test Error



(b) 95% confidence level, Differences in mean levels of variable

Figure 2: Uniform Network Results

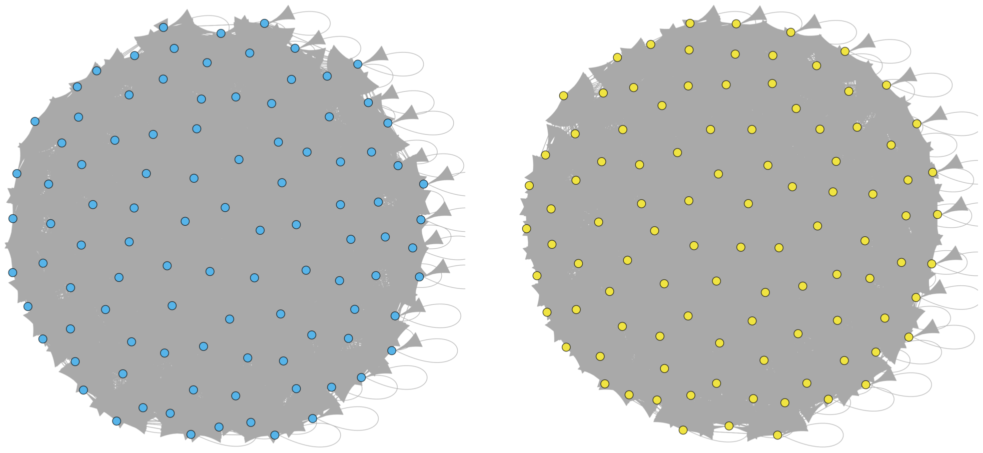


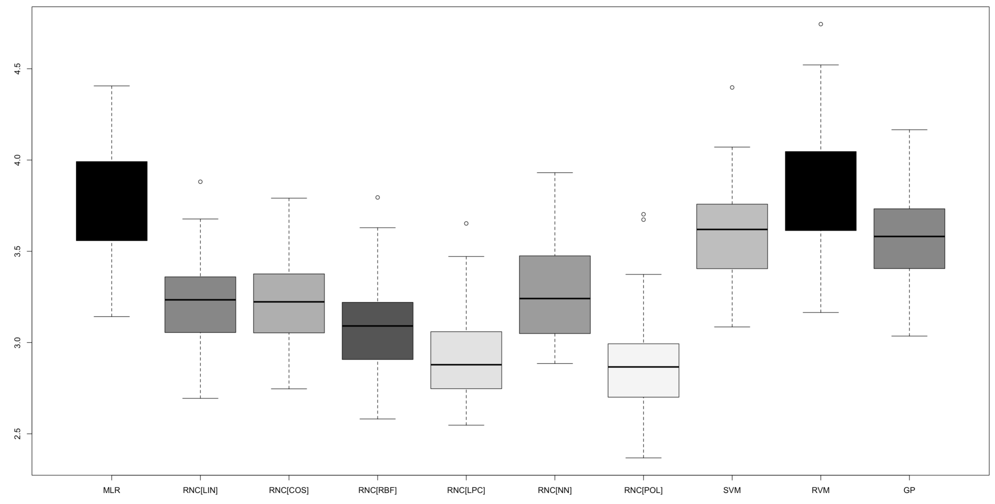
Figure 3: Bipartite Networks include two groups of networks shown in two colors.

Table 2: Mean squared training and test error with Bipartite network for Friedman1 data.

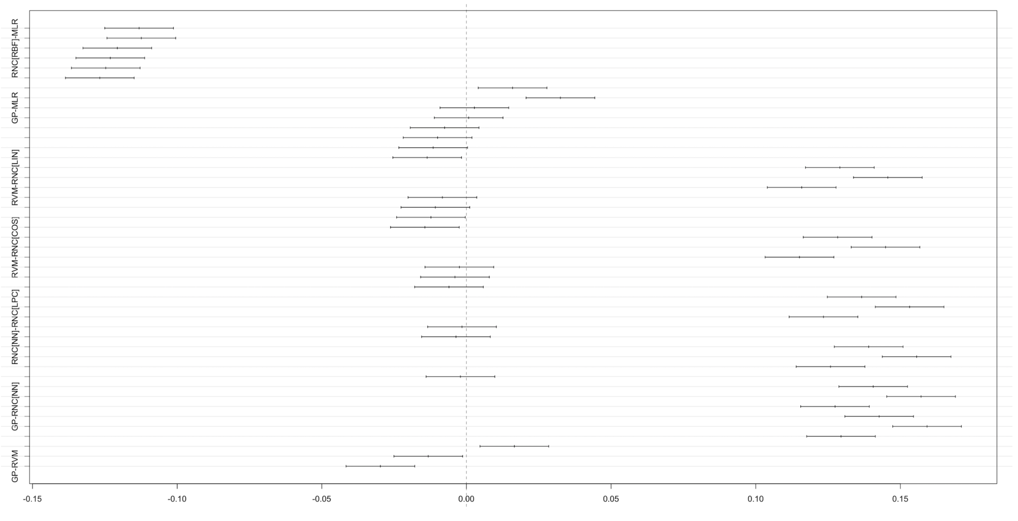
SN	MLR	LIN	COS	RBF	LPC	NN	POL	SVM	RVM	GP
Training Error	3.6	2.17	2.44	2.16	**1.42**	2.38	1.85	15.27	2.43	15.28
Test Error	3.78	3.22	3.22	3.09	2.91	3.27	**2.87**	3.6	3.84	3.57

Table 3: Mean squared training and test error with Bipartite network for Friedman3 data.

SN	MLR	LIN	COS	RBF	LPC	NN	POL	SVM	RVM	GP
Training Error	0.9907	0.1008	0.1162	0.1177	0.0845	0.1158	**0.0812**	1.679	0.914	1.6614
Test Error	1.0357	0.1539	0.1554	0.1527	0.1332	0.156	**0.1186**	1.1916	1.107	1.069

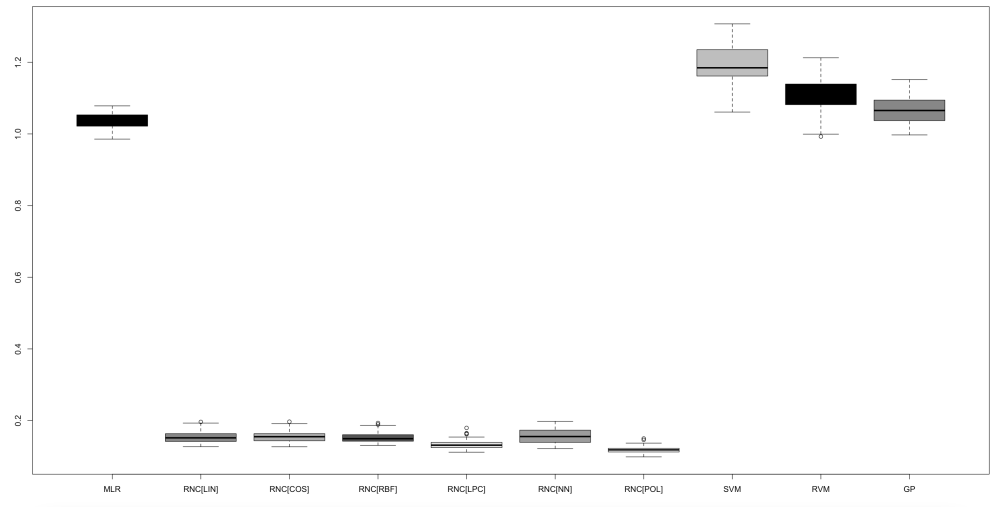


(a) Friedman1 Test Error

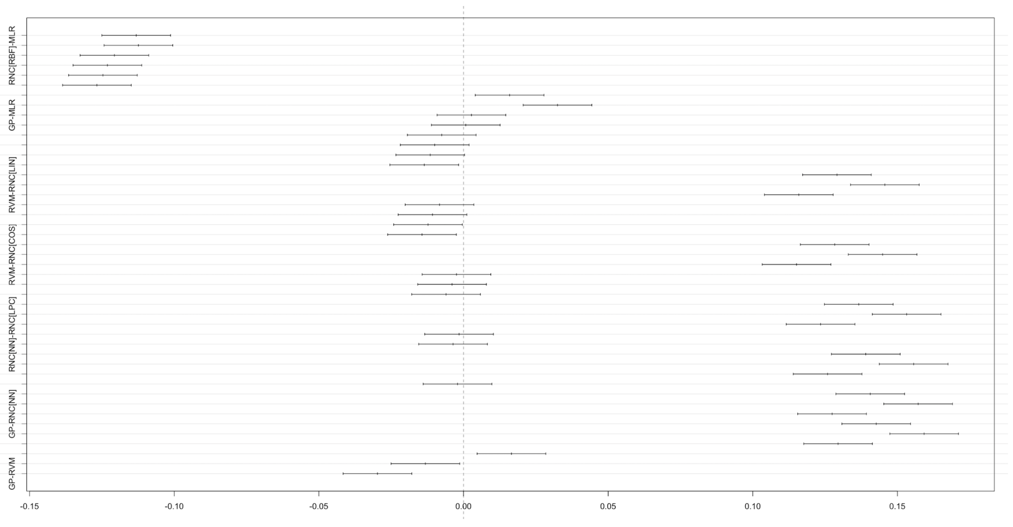


(b) 95% confidence level, Differences in mean levels of variable

Figure 4: Friedman1 Results

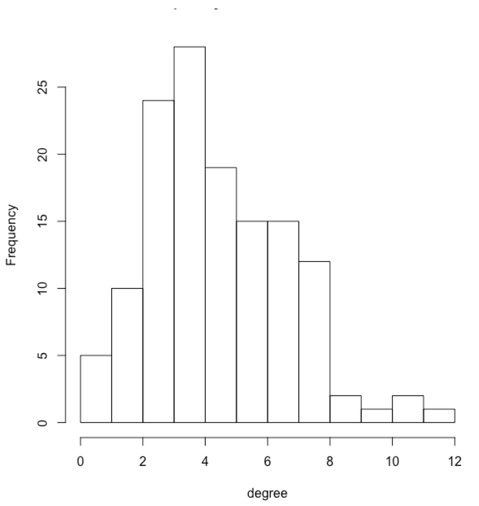


(a) Friedman3 Test Error

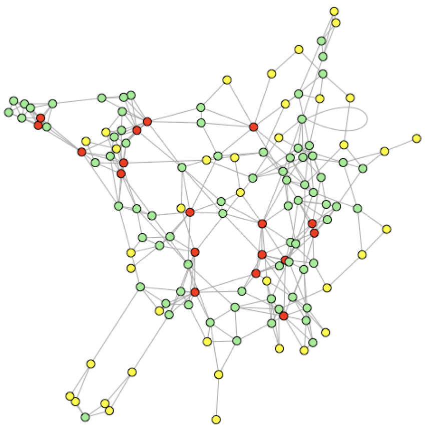


(b) 95% confidence level, Differences in mean levels of variable

Figure 5: Friedman3 Results



(a) The histogram of the numbers of friends, most students have 3-5 friends in school.



(b) The Friendship network: The network of all the students in the data we are using, there are 3 colors in the plots: yellow means one has less than 4 friends, green means one has 4-8 friends, red means one has more than 8 friends.

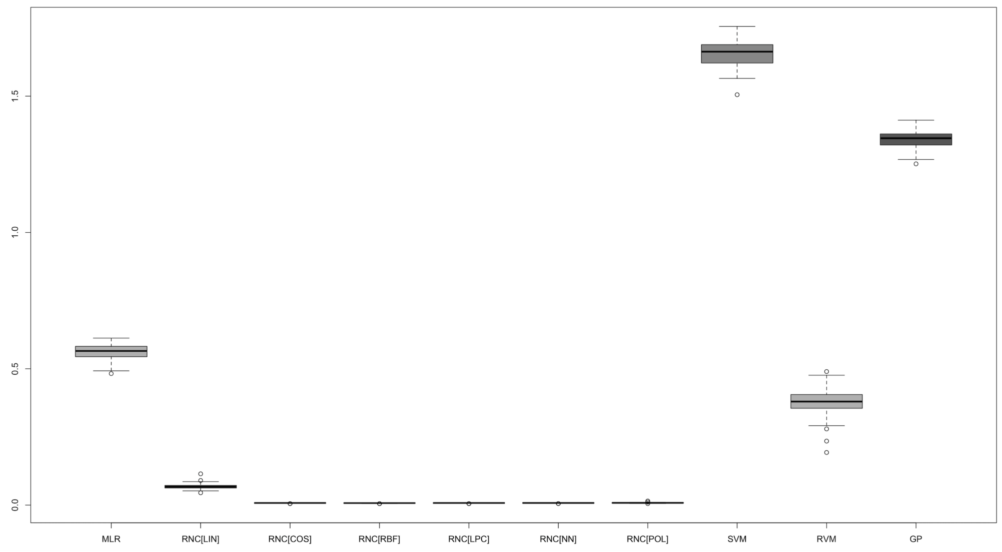
Figure 6: The Friendship Data summary

Table 4: Training error for the 'Teenage Friends, and Lifestyle Study' data.

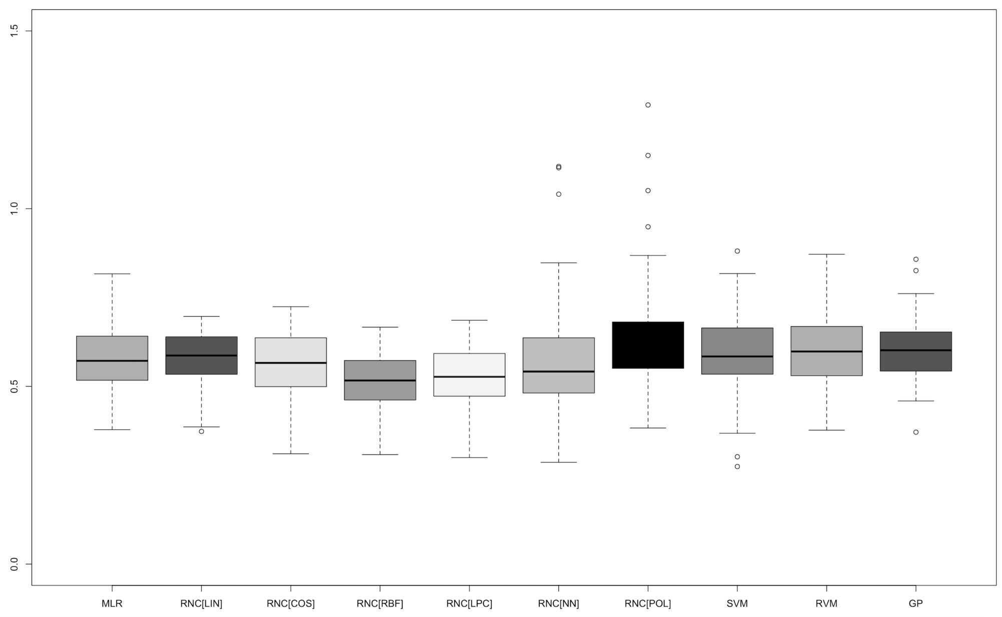
SN	MLR	LIN	COS	RBF	LPC	NN	POL	SVM	RVM	GP
Training Error	0.5617	0.0679	0.0074	**0.0071**	0.0074	0.0075	0.0081	1.6559	0.3772	1.3399

Table 5: Test error for the 'Teenage Friends, and Lifestyle Study' data.

SN	MLR	LIN	COS	RBF	LPC	NN	POL	SVM	RVM	GP
Test Error	0.5809	0.5796	0.5601	**0.5148**	0.5262	0.579	0.6688	0.5949	0.6066	0.6038



(a) Training error



(b) Test error

Figure 7: The left image shows the mean MSE for the training set. The right image shows the mean MSE for the test set.

3.4 Applications

This section evaluates our method using the *Teenage Friends and Lifestyle Study* data [7]. The social network data were collected in the Teenage Friends and Lifestyle Study. Friendship network data and substance use were recorded for a cohort of pupils in a school in the West of Scotland. The panel data were recorded over three years, starting in 1995, when the pupils were aged 13, and ending in 1997. A total of 160 pupils took part in the study. 129 pupils were present at all three measurement points. The friendship networks were formed by allowing the pupils to name up to twelve best friends. Pupils were also asked about substance use and adolescent behavior associated with, for instance, lifestyle, sporting behavior, and tobacco, alcohol, and cannabis consumption. The school was representative of others in the region in terms of social class composition.

In this application study, we focus on one question, "How often does one use Alcohol?" using eight predictor variables: age, sex, romantic involvement, family smoking habits, money (their allowance), sporting behavior, and church attendance. The data is split into training and test sets, which are 70 % and 30 %. We compare the Mean Squared Error among all the machines used in the simulation study.

According to Table 4, all models incorporating network cohesion outperform those that do not. The table 5 shows that most RNC-kernel models have a smaller MSE compared to other models, except the one with the polynomial kernel; the RNC-linear model is slightly better than MLR, and both of them are better than SVM, RVM, and GP. Among all the machines, RNC with Gaussian kernel is the best one. These results indicate that incorporating network cohesion significantly enhances predictive performance. Particularly, kernel regression with network cohesion machines shows marked improvements over linear regression with network cohesion. This suggests that selecting the appropriate kernel, as demonstrated by this experiment, is crucial for optimizing results.

4 Conclusion

The incorporation of network data into predictive models remains a critical and active area of research in statistical machine learning. In this paper, we extended the foundational work by Li et al. [1] by introducing a kernelized regression framework that accounts for both the input data and the structure of the network in a more flexible and powerful way. Our proposed method advances the state-of-the-art by addressing the limitations of the generalized linear model framework and extending it to handle nonlinear relationships, which are often present in real-world data but remain undetectable by linear models.

The primary innovation of this work is the seamless integration of graph-theoretic constraints into a kernelized regression framework. By leveraging undirected and unweighted graphs, we model relationships within the data in a straightforward but powerful manner, allowing us to incorporate social networks or other relational structures directly into the predictive model. This not only enhances interpretability but also results in significant improvements in predictive accuracy.

Through experiments using both simulated and real-world data, including the Teenage Friends and Lifestyle Study, we demonstrated the superior performance of our proposed kernel regression model over traditional linear and generalized linear models with network cohesion. The results consistently showed that the incorporation of network data, coupled with the flexibility of kernel methods, led to better generalization and more accurate predictions, particularly in scenarios where the underlying relationships were highly nonlinear.

Moreover, the kernelized approach is highly adaptable, allowing for the use of various kernel functions, such as the Gaussian Radial Basis Function (RBF), polynomial kernels, and others, depending on the specific problem at hand. This flexibility makes the method applicable to a wide range of domains where network data is prevalent, such as biology, social science, economics, and beyond.

Looking forward, there are several avenues for future work. One promising direction is the exploration of more complex graph structures, such as directed or weighted graphs, which could capture additional nuances in the relationships between entities. Furthermore, alternative regularization techniques could be investigated to improve the stability and performance of the model in cases where the network structure is particularly sparse or irregular. The development of efficient algorithms for large-scale network data is another important area of future research, particularly in the context of high-dimensional and big data applications, but also in studies where the sample size n is very large to the point of making our present treatment not scalable.

We recall that our chief motivation for this manuscript was inspired by the fact that Kernel methods are known for their flexibility and powerful ability to capture complex, nonlinear patterns. However, a major challenge arises when scaling these methods to large datasets. This is due to the fact that kernel methods revolve around the computation of the Gram matrix K, which is of size $n \times n$. Operations involving K typically have a computational complexity of $O(n^3)$, rendering kernel methods unscalable for large values of n. To address this issue, several techniques have been developed to enhance the scalability of kernel methods:

1. A common approach to reduce the computational burden is to approximate the Gram matrix. One powerful technique is the *Nyström approximation*, which approximates the full kernel matrix using a smaller subset of randomly selected data points. This reduces the computational complexity from $O(n^3)$ to $O(m^2n)$, where $m \ll n$ is the number of selected samples. Another technique is the use of *random Fourier features*, which map the data into a lower-dimensional space where linear methods can be applied to approximate the kernel trick efficiently, reducing the complexity to linear in n.

2. Another powerful approach is the use of incremental or online learning algorithms, such as *Incremental Support Vector Machines (ISVMs)* and *Incremental Gaussian Processes*. These methods allow the model to be updated incrementally as new data arrives, without the need to recompute the entire kernel matrix from scratch. This makes them particularly well-suited for streaming data or large-scale scenarios.

3. Low-rank approximations, such as *Kernel Principal Component Analysis (KPCA)*, can project the data onto a lower-dimensional subspace. This significantly reduces the dimensionality of the Gram matrix, lowering the computational and storage requirements while still capturing the essential structure of the data.

4. Finally, *distributed and parallel computing frameworks*, such as MapReduce and Apache Spark, have been employed to distribute the computation of kernel methods across multiple nodes. This approach enables efficient handling of large datasets by leveraging the power of parallel computation. While we do not implement any of the scalability approaches in this manuscript, the most natural extension of our work in the future of future work, is a straightforward adaptation of our present work to studies with large n.

By combining these strategies, kernel methods can be made more scalable, allowing them to tackle large-scale problems while maintaining their ability to capture complex, nonlinear patterns.

In conclusion, the proposed kernelized regression framework represents a substantial advancement in the modeling of network-linked data. By extending the capabilities of previous models, this work paves the way for more robust, flexible, and accurate machine learning methods that can be applied to a wide variety of complex, real-world datasets.

References

[1] T. Li, E. Levina, J. Zhu, Prediction models for network-linked data, *The Annals of Applied Statistics*, 13:132–164, 2019.

[2] M. E. J. Newman, *Networks: An Introduction*, First Edition, Oxford University Press, 2010.

[3] A. G. Marques, S. Segarra, G. Leus, A. Ribeiro, Stationary Graph Processes and Spectral Estimation, *IEEE Transactions on Signal Processing*, 65:5911–5926, 2017.

[4] Y. Shen, B. Baingana, G. B. Giannakis, Tensor Decompositions for Identifying Directed Graph Topologies and Tracking Dynamic Networks, *IEEE Transactions on Signal Processing*, 65:3675–3687, 2017.

[5] N. Meghanathan, Unit Disk Graph-Based Node Similarity Index for Complex Network Analysis, *Complexity*, 2019:6871874, 2019.

[6] A. Venkitaraman, S. Chatterjee, P. Händel, Predicting Graph Signals using Kernel Regression where the Input Signal is Agnostic to a Graph, *arXiv:1706.02191*, 2017.

[7] Excerpt of 50 girls from the *Teenage Friends and Lifestyle Study* data set, 2017, `https://www.stats.ox.ac.uk/~snijders/siena/s50_data.htm`

[8] T. Li, E. Levina, J. Zhu, C. M. Le, *randnet: Random Network Model Estimation, Selection and Parameter Tuning*, `https://cran.r-project.org/package=randnet`

[9] A. Karatzoglou, A. Smola, K. Hornik, A. Zeileis, kernlab – An S4 Package for Kernel Methods in R, *Journal of Statistical Software*, 11:1–20, 2004.

[10] L. Csató, E. Fokoué, M. Opper, B. Schottky, O. Winther, Efficient Approaches to Gaussian Process Classification, *Advances in Neural Information Processing Systems (NIPS 1999)*, 12:251–257, The MIT Press, 1999.

[11] E. Fokoué, Stable Radial Basis Function Selection via Mixture Modelling of the Sample Path, Technical Report, Rochester Institute of Technology, New York, 2009.

[12] E. Fokoue, P. Goel, The relevance vector machine: An interesting statistical perspective, Technical Report EPF-06-10-1, Kettering University, Michigan, 2006.

[13] E. Fokoué, Estimation of atom prevalence for optimal prediction, *Contemporary Mathematics*, 443:103–129, 2007.

[14] J. Quiñonero-Candela, C. E. Rasmussen, A Unifying View of Sparse Approximate Gaussian Process Regression, *Journal of Machine Learning Research*, 6:1939–1959, 2005.

[15] C. E. Rasmussen, C. K. I. Williams, *Gaussian Processes for Machine Learning*, The MIT Press, 2005.

[16] H. Drucker, C. J. C. Burges, L. Kaufman, A. Smola, V. Vapnik, Support Vector Regression Machines, *Advances in Neural Information Processing Systems (NIPS 1996)*, 9:155–161, The MIT Press, 1996.

[17] V. Vapnik, *The Nature of Statistical Learning Theory*, Second Edition, Springer Science and Business Media, 2013.

[18] M. E. Tipping, Sparse Bayesian Learning and the Relevance Vector Machine, *Journal of Machine Learning Research*, 1:211–244, 2001.

[19] G. Wahba, *Spline Models for Observational Data*, SIAM, 1990.

[20] G. Wahba, Support Vector Machines, Reproducing Kernel Hilbert Spaces and the Randomized GACV, Technical Report 984, University of Wisconsin, Madison, 1998.

[21] T. Liao, W.-Q. Wang, B. Huang, J. Xu, Learning Laplacian Matrix for Smooth Signals on Graph, *2019 IEEE International Conference on Signal, Information and Data Processing (ICSIDP)*, 2019.

[22] J. Moody, J. Coleman, Clustering and Cohesion in Networks: Concepts and Measures, *International Encyclopedia of the Social and Behavioral Sciences (Second Edition)*, pp. 906–912, 2015.

[23] C. R. Shalizi, A. C. Thomas, Homophily and Contagion Are Generically Confounded in Observational Social Network Studies, *Sociological Methods and Research*, 40:211–239, 2011.

[24] M. Vedanayaki, A Study of Data Mining and Social Network Analysis, *Indian Journal of Science and Technology*, 7:185–187, 2014.

[25] R. J. Barnes, Matrix Differentiation, Springs Journal, pp. 1–9, 2006.

[26] B. Clarke, E. Fokoue, H. H. Zhang, *Principles and Theory for Data Mining and Machine Learning*, Springer Science and Business Media, 2009.

[27] J. H. Friedman, Multivariate Adaptive Regression Splines, *The Annals of Statistics*, 19:1–67, 1991.

[28] L. Breiman, Bagging predictors, *Machine Learning*, 24:123–140, 1996.