UNIVERSITY OF CALIFORNIA,
IRVINE

Parameter Analysis on Variants of Kernel Regression over Graphs

DISSERTATION

submitted in partial satisfaction of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

in Electrical Engineering

by

Yue Zhao

Dissertation Committee:
Professor Ender Ayanoglu, Chair
Assistant Professor Yanning Shen
Professor Zhiying Wang

2023

DEDICATION

To my parents

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ALGORITHMS

ACKNOWLEDGMENTS

First, I would like to thank my advisor, Prof. Ender Ayanoglu, who is always encouraging me moving on and patient when I am facing problems. He generously shares his ideas with me and provides valuable comments and suggestions. I cannot achieve where I am without his support. I also want to thank Prof. Yanning Shen who is responsive in answering my questions regarding her work and kindly invites me to discuss related topics with her team. In addition, I want to thank Prof. Zhiying Wang who is supportive on my defense when I am in need.

I would like to thank my friends in UCI. Although we may not in the same major, we always encourages each other to continue on our academic path. I am treasuring and will always treasure our friendship wherever we are.

I would like to thank my my parents and my partner for their endless love. Without their financial and emotional support, I will never become what I am today. I am proud of them.

VITA

Yue Zhao

EDUCATION

Doctor of Philosophy in Electrical Engineering **2023**
University of California, Irvine *Irvine, CA, USA*

Master of Science in Electrical Engineering **2019**
University of California, Irvine *Irvine, CA, USA*

Bachelor of Science in Information and Electrical Engineering **2016**
Beijing Institute of Technology *Beijing, China*

RESEARCH EXPERIENCE

Graduate Research Assistant **Winter, 2020**
University of California, Irvine *Irvine, California, USA*

TEACHING EXPERIENCE

Teaching Assistant **2019–2022**
University of California, Irvine *Irvine, California, USA*

REFEREED JOURNAL PUBLICATIONS

Gaussian Kernel Variance for an Adaptive Learning Method on Signals Over Graphs **2022**

IEEE Transactions on Signal and Information Processing over Networks

ABSTRACT OF THE DISSERTATION

Parameter Analysis on Variants of Kernel Regression over Graphs

By

Yue Zhao

Doctor of Philosophy in Electrical Engineering

University of California, Irvine, 2023

Professor Ender Ayanoglu, Chair

The dissertation focuses on understanding parameter influences on variants of kernel regression models over graphs. Graphs are used to represent complex systems where components in the system are modeled as nodes, and relationships among the components are denoted as edges connecting nodes. Kernel regression models can be used to solve graph-related problems such as graph signal reconstruction and prediction. In the graph signal reconstruction problem, a common case is to predict an unknown attribute of a node using known values of the same attribute from other nodes and the network structure. In the graph prediction problem, a common case is to predict a graph signal over the network based on historical graph signals. The essence of the two problems is to model the input-output relationship, and the kernel-based regression model with an iterative solution is a simple yet possibly powerful solution. The dissertation will first show an application of the kernel regression model on the graph signal reconstruction problem over multi-layer graphs. The graph signal reconstruction problem aims to estimate unknown nodal values based on known nodal values and the multi-layer network structure. Viewing the mapping from the local network structure of a node to the nodal value as a function in a Reproducing Kernel Hilbert Space (RKHS), a regression model based on multiple kernels is built and a minimization problem is formatted. With the gradient descent algorithm, it is easy to find the solution to the minimization problem iteratively. In this application of the kernel-based regression model, the predicting

ability of the model is verified. It is also seen from the application that the single-kernel models are used as building blocks of a multi-kernel model and that the performance is dependent on the hyper-parameter settings on the single-kernel regression models. To achieve better performance with less computational cost by selecting suitable hyper-parameters for the model, the dissertation then presents an analysis framework to analyze the influence of the hyper-parameters on the predictions of single-kernel regression models. Noting that due to the iterative nature of the model solution, it is hard to figure out the influence of the hyper-parameters directly. So, the main idea of the proposed framework is to express the model prediction as a weighted sum of the training observations, and then to analyze the the influence of parameters on the observation weights. With the framework, it is found that the weights of the parameters are scaled kernel values of the input for prediction and inputs for training observations. This verifies that the kernel performs as a similarity measure and shows that the scaling factor for kernel values is related to the time difference between the two inputs of the kernel. The framework helps better understand the impact of the hyper-parameters and hints at suitable selections of those parameters. After that, the framework is generalized to do parameter analysis for an iterative solution of a kernel regression model dealing with the graph signal prediction problem where the input is agnostic. In the generalized framework, the solution acquired from the batch gradient descent algorithm can be analyzed, making the solution acquired from the gradient descent algorithm a special case.

Chapter 1

Introduction

Complex systems can be abstracted by graphs [1] which consist of nodes and edges. In those graphs, nodes represent agents while edges denote a relationship among the set of agents. For example, the citation network of a set of papers in the field of machine learning can be abstracted as a graph where papers are modeled as nodes and an edge from one node to another means that one paper cites the other paper [2]. With the ability of recording relational data, the graph structure is widely used in analyzing climate change [3, 4], pandemic evolution [5, 6], transportation conjunction [7, 8], power systems [9], and brain activity [10], to name a few.

It is very often that nodes carry features about themselves. For instance, in the aforementioned citation network [2], each paper is categorized into one of the seven predefined topics. For analytical simplicity, researchers usually use a scalar to record a feature for a node. Gathering feature values for all nodes in the network, we can form the information in a vector whose elements are indexed by nodes. The resulting vector is called a *graph signal*. To better analyze and make use of graph signals, more and more effort is put in the field of graph signal processing (GSP) [11, 12, 13, 14].

There are many practical topics related with GSP, such as graph signal compression, graph signal reconstruction [15, 16], graph signal interpolation [17], and graph signal estimation. Analogous to conventional signal processing tools, researchers interested in GSP have come up with GSP tools such as graph Fourier transform and graph filtering. The main concept is to utilize the graph structure, extracting "dominant" information and making use of it. However, most works studying such GSP tools are constrained to consider linear transformations [18]. To cope with the nonlinearity residing in signals over graphs, one way is to design non-linear graph filters [19]. Another way to consider the nonlinearity is to apply kernel methods on graphs [20, 21, 22]. Nonlinear algorithms usually outperform their linear counterparts, however, their computational costs also grow faster with more known nodes, making them less practical for large networks.

Fortunately, [23] discovers that for shift-invariant kernels, the kernel value can be approximated using random features drawn from the Fourier transform of the kernel. Thanks to this discovery, kernel-based models could have efficient solutions. In this dissertation, we study kernel-based regression models related to graphs with random Fourier features (KRG-RFF) since they are simple yet powerful with proper configuration. Prior to the detailed parameter analysis framework for KRG-RFF, we will first go over basic backgrounds related with KRG-RFF, then, we will present an application of KRG-RFF to show the predicting ability of KRG-RFF models.

1.1 Notations

Vectors are denoted by bold lowercase characters, and matrices are denoted by bold uppercase characters. Sets are denoted in a calligraphic font. The (m, n)-th (m-th) element in an $M \times N$ matrix $\mathbf{A}$ ($M \times 1$ column vector or $1 \times M$ row vector $\mathbf{a}$) is denoted by $[\mathbf{A}]_{m,n}$ ($\mathbf{a}_m$) where $1 \leq m \leq M$ and $1 \leq n \leq N$. The symbol $\mathbf{1}_K$ represents the vector with a

length of K whose elements are all 1. Similarly, $\mathbf{0}_{M\times N}$ represents the matrix with size of $M \times N$ whose elements are all 0. The operation $\text{diag}(\mathbf{a})$ generates a diagonal matrix whose (i,i)-th element is equal to $[\mathbf{a}]_i$. And the operation $\text{vec}(\mathbf{A})$ generates a column vector by stacking columns of $\mathbf{A}$. The maximum (minimum) element in vector $\mathbf{a}$ is represented by $\max(\mathbf{a})$ ($\min(\mathbf{a})$), while $\max_{\mathbf{a}} J(\mathbf{a})$ ($\min_{\mathbf{a}} J(\mathbf{a})$) represents the maximum (minimum) value of the objective function $J(\mathbf{a})$ for all applicable $\mathbf{a}$. The symbol $(\cdot)^\top$ denotes the transpose operation. The l_1 norm, l_2 norm, and Frobenius norm are denoted by $\|\cdot\|_1$, $\|\cdot\|_2$, and $\|\cdot\|_F$, respectively. The notation $|\cdot|$ represents the absolute value for a number, or the cardinality of a set. The (conditional) expectation is denoted by $\mathbb{E}[\cdot]$ ($\mathbb{E}[\cdot|\text{given variable}]$), and the (conditional) variance is denoted by $\mathbb{V}[\cdot]$ ($\mathbb{V}[\cdot|\text{given variable}]$). The Fourier transform is denoted by $\mathcal{F}$. The normal distribution is denoted by $\mathcal{N}$ and the uniform distribution is denoted by $\mathcal{U}$. For convenience in reading, all notations are summarized in Table 1.1.

1.2 KRG-RFF

In this section, we first review basic concepts of graph structure, graph signal, and smoothness. Then we refer to the original KRG algorithm in its first proposed form with a closed-form solution. After that, random Fourier feature (RFF) approximation is reviewed, paving the way to KRG-RFF at the end of the section.

1.2.1 Graph Structure

Consider a weighted, undirected graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}, \mathcal{W}\}$ where $\mathcal{N}$ is the set of nodes with $|\mathcal{N}| = K$, $\mathcal{E}$ is the set of edges, and $\mathcal{W}$ contains weights for all edges. The graph structure can be expressed in the form of the *adjacency matrix* $\mathbf{A}$ of size $K \times K$. More specifically,

Table 1.1: Notations and their meanings.

Notation	Meaning
$\mathbf{a}$	a vector
$\mathbf{A}$	a matrix
$\mathcal{S}$	a set
$[\mathbf{a}]_m$	m-th element of $\mathbf{a}$
$[\mathbf{A}]_{m,n}$	(m,n)-th element of $\mathbf{A}$
$[\mathbf{1}]_K$	all-one vector with length K
$[\mathbf{0}]_{M\times N}$	all-zero matrix with size $M \times N$
$\text{diag}\,(\mathbf{a})$	the diagonal matrix with $\mathbf{a}$ at its diagonal
$\text{vec}\,(\mathbf{A})$	the vectorization of $\mathbf{A}$
$\max(\mathbf{a})$	the maximum element in $\mathbf{a}$
$\max_{\mathbf{a}} J(\mathbf{a})$	the maximum value for $J(\mathbf{a})$
$\min(\mathbf{a})$	the minimum element in $\mathbf{a}$
$\min_{\mathbf{a}} J(\mathbf{a})$	the minimum value for $J(\mathbf{a})$
$(\cdot)^\top$	the transpose operation
$\|\cdot\|_1$	the l_1 norm
$\|\cdot\|_2$	the l_2 norm
$\|\cdot\|_F$	the Frobenius norm
$\|\cdot\|$	the absolute value or the cardinality
$\mathbb{E}[\cdot]$ ($\mathbb{E}[\cdot\|$given variable$]$)	(conditional) expectation
$\mathbb{V}[\cdot]$ ($\mathbb{V}[\cdot\|$given variable$]$)	(conditional) variance
$\mathcal{F}\{f\}$	the Fourier transform of the function f
$\mathcal{N}(a,b)$	the normal distribution with mean a and variance b
$\mathcal{U}(a,b)$	the uniform distribution over the interval (a,b)

elements of $\mathbf{A}$ are defined as

$$[\mathbf{A}]_{k,l} = \begin{cases} w_{k,l} & \text{if } (v_k, v_l) \in \mathcal{E} \\ 0 & \text{otherwise} \end{cases}$$

where $v_k, v_l \in \mathcal{N}$ and $w_{k,l}$ is the weight on the edge (v_k, v_l). Since we are interested in undirected graphs, we have $\mathbf{A}^\top = \mathbf{A}$. Furthermore, we focus on simple graphs that contain no self-loops and have non-negative weights. That is, the diagonal elements of $\mathbf{A}$ are all zeros, and $[\mathbf{A}]_{k,l} \geq 0, \forall k, l \in \{1, 2, \cdots, K\}$. Note that unweighted graphs can be viewed as a special case of weighted graphs where all edge weights are equal to 1.

1.2.2 Graph Signal & Smoothness

In some cases, we are interested in node feature values in terms of some aspect. Suppose the feature value of node v_k is t_k, then we can get $\mathbf{t} = [t_1, t_2, \cdots, t_K]^\top$ which is called a graph signal. In practice, a property is commonly observed that connected nodes tend to have closer feature values [24]. That means graph signals are supported by underlying structures. When using the relevant structure, this property of a graph signal can be quantified as smoothness [24] which is defined as

$$\text{smoothness}_{\mathbf{A}}(\mathbf{t}) = \frac{1}{2} \sum_{k,l \in \{1,\cdots,K\}} [\mathbf{A}]_{k,l} \left(t_k - t_l\right)^2 . \tag{1.1}$$

The right-hand side of (1.1) shows that the smoothness value is small only when the nodal value difference for nodes connected by large-weight edges is small. A small smoothness value implies the graph signal is smooth over the graph structure. By defining $\mathbf{L} = \text{diag}\left(\mathbf{A1}\right) - \mathbf{A}$, we can rewrite (1.1) as

$$\text{smoothness}_{\mathbf{L}}(\mathbf{t}) = \mathbf{t}^\top \mathbf{L} \mathbf{t}.$$

Note that $\mathbf{L}$ is the well-known combinatorial graph Laplacian matrix [14]. It plays an important role in GSP tools such as the graph Fourier transform [17], graph filtering [25], and community detection [26].

1.2.3 KRG

The kernel-based regression model over graphs (KRG) predicts graph signals out of agnostic inputs. Denote the input by $\mathbf{x}$ and the output by $\mathbf{y} = [y_1, \cdots, y_K]^\top$. Note that the output $\mathbf{y}$ is a graph signal whereas the input $\mathbf{x}$ is not necessarily a graph signal. The input $\mathbf{x}$ can be any vector that is helpful in predicting $\mathbf{y}$. The KRG-RFF model assumes input-output

pairs $(\mathbf{x}, y_k), \forall k \in \{1, \cdots, K\}$ originate from a reproducing kernel Hilbert space (RKHS) [27]. With the representer theorem [28], the input-output relationship can be expressed as

$$\mathbf{y} = \sum_{n=1}^{N} \Psi_n \kappa(\mathbf{x}_n, \mathbf{x}) = \mathbf{W}^\top \phi(\mathbf{x}) \tag{1.2}$$

where κ is a pre-selected kernel with the kernel trick [29] $\kappa(\mathbf{x}_m, \mathbf{x}_n) = \phi(\mathbf{x}_m)^\top \phi(\mathbf{x}_n)$, $\phi(\cdot)$ denotes the transformation from the input space to the feature space of the kernel κ, and $\mathbf{x}_n$ is the input at time n. The total number of input-output pairs $(\mathbf{x}_n, \mathbf{t}_n)$ used for training is denoted by N. Then, the goal is to minimize the following objective function [30]

$$C(\mathbf{W}) = \sum_{n=1}^{N} \|\mathbf{t}_n - \mathbf{y}_n\|_2^2 + \alpha \operatorname{tr}\left(\mathbf{W}^\top \mathbf{W}\right) + \beta \sum_{n=1}^{N} \mathbf{y}_n^\top \mathbf{L} \mathbf{y}_n \tag{1.3}$$

where $\alpha > 0$ and $\beta > 0$ are pre-selected parameters based on applications. The first term on the right-hand side of (1.3) is the total squared error between graph signal predictions and their ground truth. The second term restricts the absolute values of elements in the model parameter $\mathbf{W}$ from being excessively large. The third term is the total smoothness value of all predicted graph signals, enforcing predictions to be smooth graph signals over known graph structure $\mathbf{L}$. Authors of [30] provide a closed-form solution as

$$\operatorname{vec}(\mathbf{\Phi}) = (\mathbf{G} + \mathbf{C})^{-1} \operatorname{vec}(\mathbf{T}) \tag{1.4}$$

by defining

$$\begin{aligned}
\mathbf{T} &= [\mathbf{t}_1, \cdots, \mathbf{t}_N]^\top \\
\mathbf{\Phi} &= [\phi(\mathbf{x}_1), \cdots, \phi(\mathbf{x}_N)]^\top \\
\mathbf{K} &= \mathbf{\Phi}^\top \mathbf{\Phi} \\
\mathbf{G} &= \mathbf{I}_K \otimes (\mathbf{K} + \alpha \mathbf{I}_N)
\end{aligned}$$

$$\mathbf{C} = \beta \mathbf{L} \otimes \mathbf{K}$$

where $\mathbf{I}_K$ is the identity matrix with size $K \times K$ and $\otimes$ denotes the Kronecker product [31].

1.2.4 RFF Approximation

For a shift-invariant kernel, we have $\kappa(\mathbf{x}_m, \mathbf{x}_n) = \kappa(\mathbf{x}_m - \mathbf{x}_n, \mathbf{0})$. For simplicity, we denote the shift-invariant kernel $\kappa(\mathbf{x}_m - \mathbf{x}_n, \mathbf{0})$ by $\kappa(\mathbf{x}_m - \mathbf{x}_n)$. With proper normalization such that $\kappa(\mathbf{x} - \mathbf{x}) = \kappa(\mathbf{0}) = 1$, authors of [23] found the random Fourier feature (RFF) approximation as

$$\kappa(\mathbf{x}_m, \mathbf{x}_n) = \kappa(\mathbf{x}_m - \mathbf{x}_n) = z^\top(\mathbf{x}_m) z(\mathbf{x}_n) \tag{1.5}$$

where $z : \mathbb{R}^K \longrightarrow \mathbb{R}^D$ is a nonlinear mapping with the form

$$z(\mathbf{x}) = \left(\frac{D}{2}\right)^{-\frac{1}{2}} \left[\cos\left(\mathbf{v}_1^\top \mathbf{x} + b_1\right), \cdots, \cos\left(\mathbf{v}_D^\top \mathbf{x} + b_D\right)\right]^\top \tag{1.6}$$

where $\{\mathbf{v}_d\}_{d=1}^D$ are random features (RFs) drawn from $p(\mathbf{v}) = \frac{\mathcal{F}\{\kappa(\mathbf{x}_m - \mathbf{x}_n)\}}{2\pi}$ and $\{b_d\}_{d=1}^D$ are drawn from $\mathcal{U}(0, 2\pi)$.

1.2.5 KRG-RFF

The solution in (1.4) needs the inversion of an $NK \times NK$ matrix whose computational cost could become prohibitively high for large networks or abundant historical observations. KRG-RFF [32] assumes the input-output relationship in (1.2), but limits the kernel to be shift-invariant. Then, the RFF approximation could be applied, and the system model

becomes

$$\mathbf{y} = \sum_{n=1}^{N} \Psi_n \kappa\left(\mathbf{x}_n, \mathbf{x}\right) = \sum_{n=1}^{N} \Psi_n z^{\top}\left(\mathbf{x}_n\right) z\left(\mathbf{x}\right) = \mathbf{H}^{\top} z\left(\mathbf{x}\right) \tag{1.7}$$

by defining $z(\cdot)$ in (1.6) and denoting $\mathbf{H}^{\top} = \sum_{n=1}^{N} \Psi_n z^{\top}\left(\mathbf{x}_n\right)$. The objective function is

$$C\left(\mathbf{H}\right) = \sum_{n=1}^{N} \left\|\mathbf{t}_n - \mathbf{y}_n\right\|_2^2 + \alpha \operatorname{tr}\left(\mathbf{H}^{\top}\mathbf{H}\right) + \beta \sum_{n=1}^{N} \mathbf{y}_n^{\top}\mathbf{L}\mathbf{y}_n. \tag{1.8}$$

The function in (1.8) is convex and thus can be solved via methods like gradient descent. Authors of [32] propose a batch-based gradient descent algorithm in which $\mathbf{H}$ is initialized as $\mathbf{0}_{D\times K}$ and updated as

$$\mathbf{H}_{n+1} = \left(1 - \mu\alpha\right)\mathbf{H}_n + \frac{\mu}{N_b}\mathbf{Z}_n^{\top}\left(\mathbf{E}_n - \beta\mathbf{Y}_n\mathbf{L}\right) \tag{1.9}$$

if n is a multiple of δ, and otherwise

$$\mathbf{H}_{n+1} = \mathbf{H}_n. \tag{1.10}$$

In (1.9) and (1.10), δ controls how frequently $\mathbf{H}$ is updated. The parameters μ and N_b are the learning rate and the batch size, respectively. And, $\mathbf{Z}_n$, $\mathbf{T}_n$, $\mathbf{Y}_n$, and $\mathbf{E}_n$ are defined as

$$\begin{aligned}
\mathbf{Z}_n &= \left[z\left(\mathbf{x}_{n-N_b+1}\right), z\left(\mathbf{x}_{n-N_b+2}\right), \cdots, z\left(\mathbf{x}_n\right)\right]^{\top} \\
\mathbf{T}_n &= \left[\mathbf{t}_{n-N_b+1}, \cdots, \mathbf{t}_n\right]^{\top} \\
\mathbf{Y}_n &= \mathbf{Z}_n\mathbf{H}_n \\
\mathbf{E}_n &= \mathbf{T}_n - \mathbf{Y}_n.
\end{aligned} \tag{1.11}$$

1.3 An Example: Multi-Layer Gradraker

This section presents an application of a graph-adaptive random-feature- and multi-kernel-based learning approach, abbreviated Gradraker [33]. Gradraker is a kind of KRG-RFF algorithm. It was proposed to perform graph signal reconstruction on single-layer graphs. In practice, due to reasons like privacy concerns, it is possible that not all values for nodes in a network are accessible. Gradraker is a solution to the problem of predicting unknown nodal values using nodes whose values are known and the network structure. Notice that Gradraker is originally expected to work on single-layer graphs, however, there are chances in reality that more than one aspect of node attributes are available and thus a multi-layer graph is generated. Details of Gradraker can be found in [33] and will be reviewed in Chapter 2. Here, we will show examples of how Gradraker could be extended to multi-layer cases to show potential usages of KRG-RFF algorithms.

1.3.1 Experts Combination

Suppose we have an L-layer graph. Each layer has the same set of vertices but with different connections, and thus there are different adjacency matrices $\mathbf{A}_l, l \in \{1, ..., L\}$ with the same size $N \times N$. For each layer, using single-layer Gradraker (Gradraker in its originally proposed form), we could get a prediction, denoted as $\hat{f}_l$ for the l-th layer. If we view individual layer predictions as Experts and final prediction as the Learner, the problem will be a prediction game with expert advice in [34]. With the Aggregating Algorithm (AA), we can combine predictions of those experts, i.e., layer predictions, by introducing β_l as the layer weight which is updated as

$$\beta_{t+1} = \beta_t \exp(\eta_t'' \mathcal{L}_t(\hat{f}_l(\mathbf{a}_{l,n}), x_n))$$

and normalized as

$$\bar{\beta}_l = \beta_l / \sum_{j=1}^{L} \beta_j$$

for $l = 1, ..., L$ where η_t'' is the preselected learning rate for layer weights at time t, $\mathcal{L}$ is the used loss function, $\mathbf{a}_{l,n}$ is the adjacency vector of node v_n in l-th layer and x_n is the ground truth for v_n. Then, the final prediction is acquired as $\hat{f}(v_n) = \sum_{l=1}^{L} \beta_l \hat{f}_l(\mathbf{a}_{l,n}), v_n \in \mathcal{V}$.

We can also combine the single-layer single-kernel predictions from the same kernel with the AA algorithm first to get a kernel prediction. Viewing kernel predictions as experts, we can apply the AA algorithm again to get the final result. Notice that the mathematical model behind this is exactly the same as the one illustrated in the last paragraph. We call these two combination methods with experts "Experts Combination" whose graphical illustrations are shown in Fig. 1.1 and Fig. 1.2. In the two figures, the first column denotes different layers of the multi-layer graph and the second column denotes the single-layer single-kernel Gradraker (submodel) predictions. Depending on different combining orders, the third column denotes layer predictions or kernel predictions.

1.3.2 Long Vector Combination

The original single-layer single-kernel Gradraker is based on $\hat{f} = \boldsymbol{\theta}^\top z(\mathbf{a})$, and the single-layer multi-kernel Gradraker can be written as $\hat{f} = \sum_{p=1}^{P} \bar{\omega}_p \boldsymbol{\theta}_p^\top z_p(\mathbf{a})$ where P is the number of kernels in the kernel dictionary. Since both ω_p and $\boldsymbol{\theta}_p$ are trainable, we view $\omega_p \boldsymbol{\theta}_p^\top$ as a whole, and denote it as $\boldsymbol{\theta}_{rev,p}$. Then we get

$$\hat{f}(\mathbf{a}) = \sum_{p=1}^{P} \boldsymbol{\theta}_{rev,p}^\top z_p(\mathbf{a}) = \boldsymbol{\zeta}^\top z_{rev}(\mathbf{a}) \tag{1.12}$$

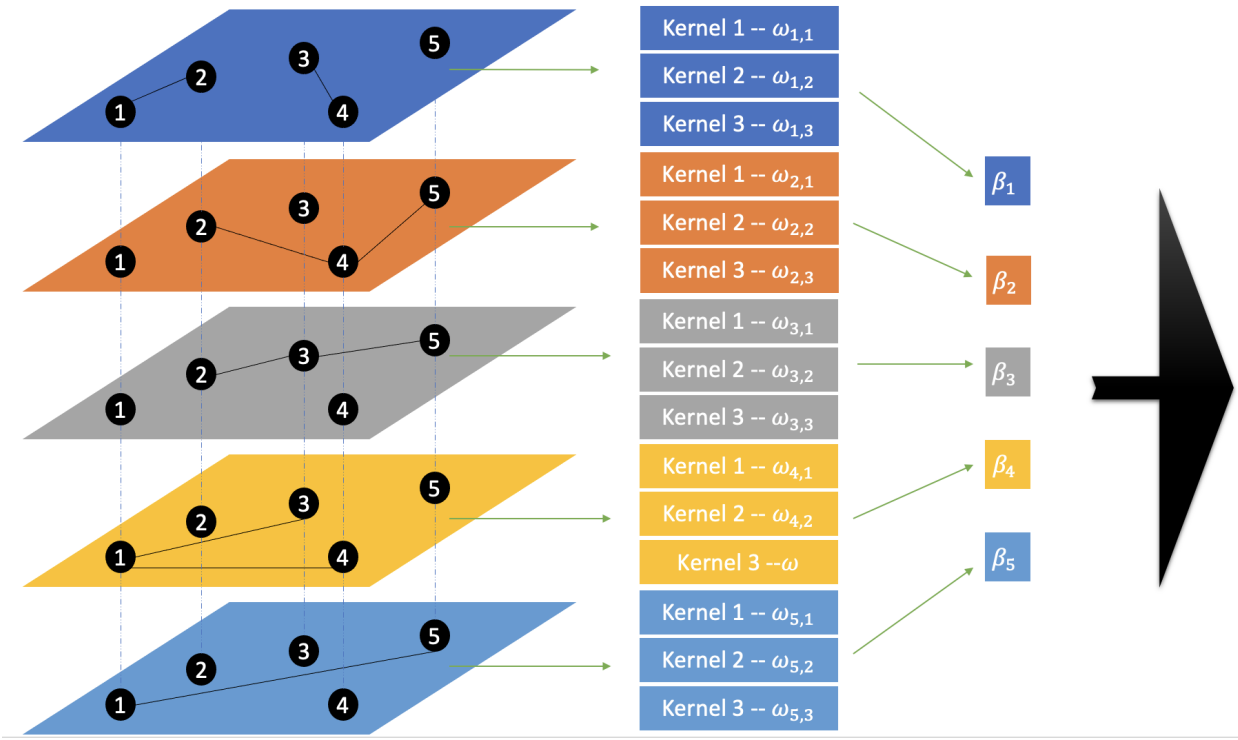

Figure 1.1: A graphic illustration of one the the Experts Combination method, the layer-first combination.

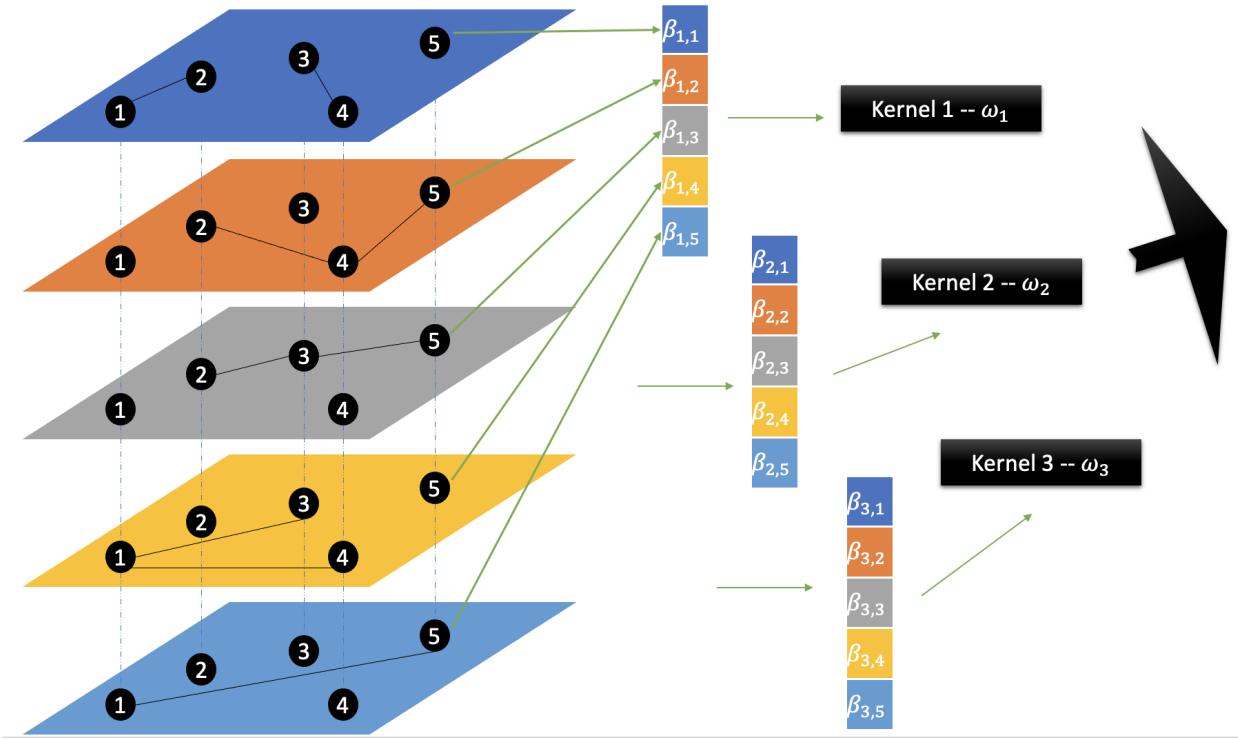

Figure 1.2: A graphic illustration of one the the Experts Combination method, the kernel-first combination.

Algorithm 1 Long Vector Combination based on Gradraker

Input: Kernels $\kappa_p, p = 1, ..., P$, step size $\eta > 0$ and number of RFs D, L-layer graph adjacency matrices and labels of known nodes
Initialization: $\boldsymbol{\upsilon}_1 = \mathbf{0}$ and P sets of RFs $\{\boldsymbol{\xi}_{p,i}\}_{i=1}^{D}$
Training:
for $t = 1, ..., T$ **do**
  $z_{revised}(v_{n_t}) = []$;
  for $l = 1, ..., L$ **do**
    for $p = 1, ..., P$ **do**
      Obtain the adjacency vector $\mathbf{a}_{l,n_t}$ of node v_{n_t} and construct $z_p(\mathbf{a}_{l,n_t})$ via (**??**);
      Update $z_{revised}(v_{n_t})$ such that $z_{revised}(v_{n_t}) = [z_{revised}(v_{n_t}); z_p(\mathbf{a}_{l,n_t})]$;
    end for
  end for
  Predict $\hat{f}(v_{n_t}) = \boldsymbol{\zeta}_t^\top z_{revised}(v_{n_t})$ and get loss $\mathcal{L}_t(\hat{f}(v_{n_t}))$ via (1.18);
  Update $\boldsymbol{\upsilon}_{t+1}$ via (1.17);
end for
Inference:
$z_{revised}(v_{new}) = []$;
for $l = 1, ..., L$ **do**
  for $p = 1, ..., P$ **do**
    Obtain the adjacency vector $\mathbf{a}_{l,new}$ of node v_{new} and construct $z_p(\mathbf{a}_{l,new})$ via (**??**);
    Update $z_{revised}(v_{new})$ such that $z_{revised}(v_{new}) = [z_{revised}(v_{new}); z_p(\mathbf{a}_{l,new})]$;
  end for
end for
Predict $\hat{f}(v_{new}) = \boldsymbol{\upsilon}_{T+1}^\top z_{revised}(v_{new})$;
If y_{new} is available, update $\boldsymbol{\upsilon}_{T+1}$ via (1.17).

where

$$z_{rev}(\mathbf{a}) = [z_1(\mathbf{a}), ..., z_P(\mathbf{a})]^\top \tag{1.13}$$

which is of size $(2D \times P) \times 1$. Note that $\boldsymbol{\zeta}$ is a trainable parameter.

Then the multi-layer multi-kernel Gradraker can be represented as $\hat{f}(v_n) = \sum_{l=1}^{L} \beta_l \boldsymbol{\zeta}_l^\top z_{rev}(\mathbf{a}_{l,n})$. Similarly, we view $\beta_l \boldsymbol{\zeta}_l^\top$ as a whole since β_l and $\boldsymbol{\zeta}_l^\top$ are trainable, and denote it as $\boldsymbol{\zeta}_{rev,l}^\top$. Then we get

$$\hat{f}(v_n) = \sum_{l=1}^{L} \boldsymbol{\zeta}_{rev,l}^\top z_{rev}(\mathbf{a}_{l,n}) = \boldsymbol{\upsilon}^\top z_{revised}(v_n) \tag{1.14}$$

where

$$z_{revised}(v_n) = [z_{rev}(\mathbf{a}_{1,n}), ..., z_{rev}(\mathbf{a}_{L,n})]^\top \tag{1.15}$$

which is of size $(2D \times P \times L) \times 1$. Note that $\boldsymbol{v}$ is a trainable parameter. Then, the problem is to solve

$$\arg\min_{\boldsymbol{v}} \frac{1}{N} \sum_{i=1}^{N} \mathcal{C}(\boldsymbol{v}^\top z_{revised}(v_i), x_i) + \mu \|\boldsymbol{v}\|^2 \tag{1.16}$$

which can also be dealt with online gradient descent as follows

$$\boldsymbol{v}_{t+1} = \boldsymbol{v}_t - \eta_t \nabla_{\boldsymbol{v}} \mathcal{L}_t \left(\boldsymbol{v}^\top z_{revised}(v), x\right) \tag{1.17}$$

where

$$\mathcal{L}_t \left(\boldsymbol{v}^\top z_{revised}(v, x)\right) = \mathcal{C} \left(\boldsymbol{v}^\top z_{revised}(v), x\right) + \mu \|\boldsymbol{\theta}\|^2. \tag{1.18}$$

We call the resulting multi-layer Gradraker variant "Long Vector Combination" and provide a graphic illustration in Fig. 1.3. The algorithm is summarized in Algorithm 1. If we start from the multi-layer single-kernel Gradraker, we will still reach at the same algorithm.

1.3.3 Simulation Results

The simulation results of the two multi-layer Gradraker variations together with a layer-aware kernel-based algorithm published in [35] on a temperature dataset of the weather stations in Switzerland during 1961-1990 [36] and 1981-2010 [37] is shown in Fig. 1.4 . The temperature dataset contains average monthly temperature from 93 weather stations during 1961-1990 and average temperature from 91 weather stations during 1981-2010 in Switzerland. We

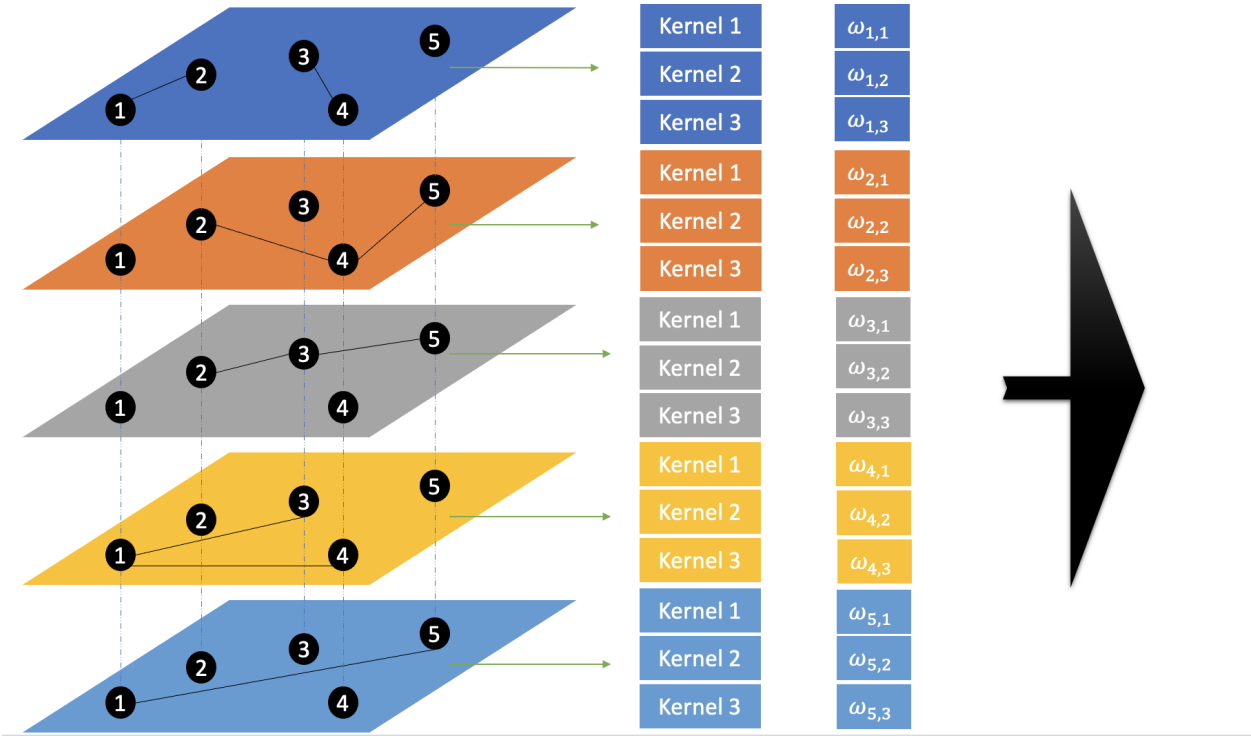

Figure 1.3: A graphic illustration of the Long Vector Combination method.

take the overlap of these two sets of stations and get 83 available stations. Information from 1961-1990 is used for training, while information from 1980-2010 is used for testing. The network structure of the 83 stations is created using the GL-SigRep algorithm in [24]. The generated network is randomly divided into 5 disjoint subgraphs, each being a single layer.

In the experiment, M nodes are randomly selected as knowns while the rest is for testing. Normalized Mean Square Error for tested nodes (Generalization NMSE) is calculated via Generalization NMSE$= \|\mathbf{y} - \mathbf{y}_{pred}\|^2 / \|\mathbf{y}\|^2$, where $\mathbf{y}$ is the true value and $\mathbf{y}_{pred}$ is the prediction. The kernel dictionary consisting of 3 Gaussian kernels with $\sigma_1^2 = 1, \sigma_2^2 = 5, \sigma_3^2 = 10$ are used for two variants, and hyperparameters for the referenced algorithm are tuned through trial and error. The parameter D is set to be 100, and the Generalization NMSE is calculated over 100 repeated experiments. Fig. 1.4 shows the (averaged) Generalization NMSE curve with respect to the number of known nodes (M) for the temperature datasets. It is seen that both the multi-layer Gradraker variants beat the referenced algorithm which is

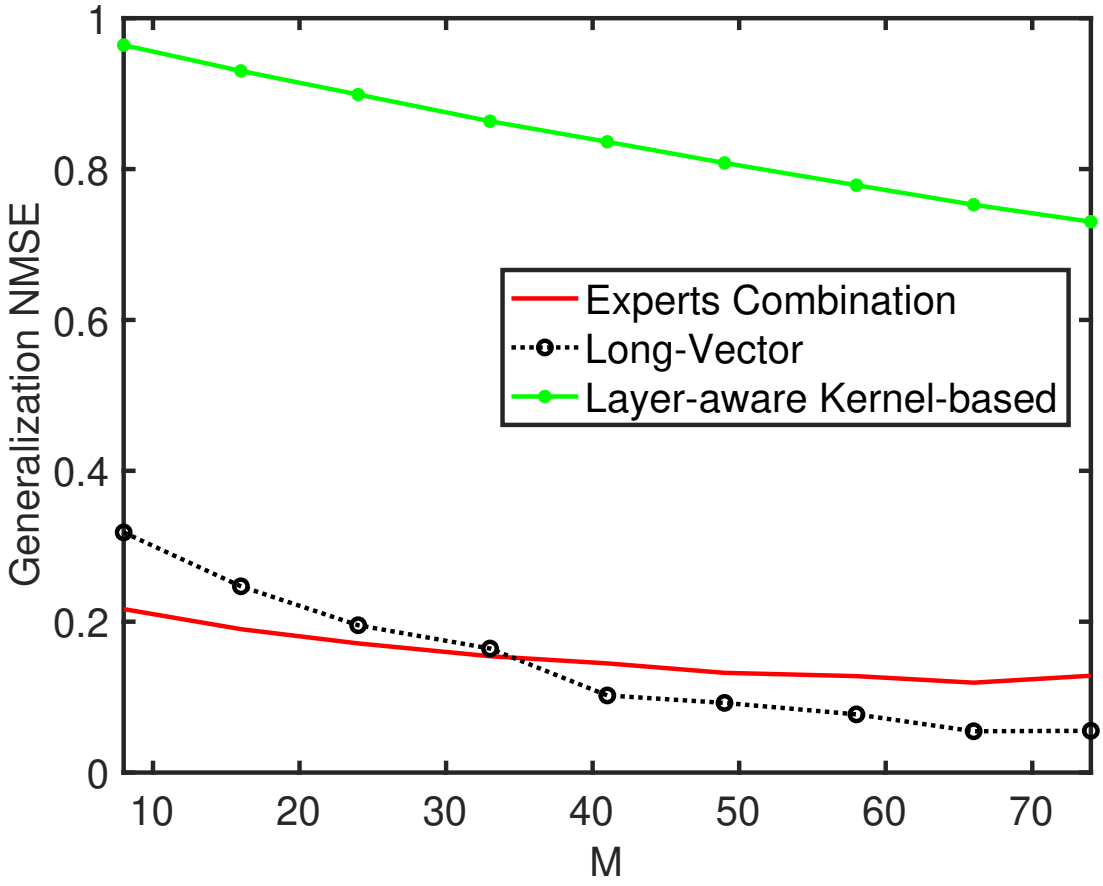

Figure 1.4: Performance on Switzerland stations' temperature dataset.

graphical-kernel-based. The predicting ability of KRG-RFF models is powerful with proper configuration.

1.4 Contributions

A Gradraker model consists of several single-kernel Gradraker (SKG) models which share the input of the Gradraker model. Each component SKG submodel outputs an estimation which is then used for acquiring the final estimation of the Gradraker model via an aggregation algorithm. That is, SKG models are building blocks of Gradraker-based algorithms and Gradraker performance is highly influenced by the best performance among all the

components.

Noticing that the computational cost grows linearly with the number of component single-layer single-kernel Gradraker submodels, when inappropriate submodels are inside, computational complexity grows beyond what is needed with appropriate submodels. So, distinguishing suitable submodels helps achieve the best performance with the lowest cost, and understanding SKG performance in a detailed manner is of great importance for future studies. In this dissertation, the contributions are:

- Establishing a parameter analysis framework to understand influences of hyper parameters in KRG-RFF models;
- Using Gaussian kernel on a KRG-RFF algorithm, the Gradraker algorithm, as an example to understand the influence of kernel parameters based on the established framework;
- Proposing suitable configurations for learning rates and kernel parameters in KRG-RFF models.

The rest of the dissertation is organized as follows. Chapter 2 introduces a simple version of the parameter analysis framework using Gaussian kernel in a sequentially trained KRG-RFF model with scalar outputs. Influence of the Gaussian kernel variance will be discussed in detail and a method of finding a suitable Gaussian variance is proposed. Then, in Chapter 3, the simple framework will be extended to deal with batch-based KRG-RFF learning models with vector outputs. Based on the framework, influences of hyper parameters, such as batch size and learning rates, are discussed. Corresponding simulations are provided in respective chapters. The dissertation will be concluded in Chapter 4.

Chapter 2

Analysis for Gaussian Variance

This chapter discusses a special kind of a simple yet possibly powerful algorithm, called single-kernel Gradraker (SKG), which is an adaptive learning method predicting unknown nodal values in a network using known nodal values and the network structure. We aim to find out how to configure the special kind of the model in applying the algorithm. To be more specific, we focus on SKG with a Gaussian kernel and specify how to find a suitable variance for the kernel. To do so, we introduce two variables with which we are able to set up requirements on the variance of the Gaussian kernel to achieve (near-) optimal performance and can better understand how SKG works. Simulation results on real datasets are provided. The rest of the chapter is organized as follows. Section 2.1 provides a brief review of the interested algorithm, Gradraker, and Section 2.2 gives a review of SKG. Mathematical tools will be introduced in Section 2.3 followed by the impact analysis of different Gaussian kernels and an algorithm to find a suitable Gaussian kernel for a given training set in Section 2.4. Section 2.5 shows simulation results to verify properties of introduced variables and effectiveness of the proposed algorithm. Discussions are provided in Section 2.6.

2.1 Introduction

Shen *et al.* [33] propose a simple yet possibly powerful algorithm called **gr**aph-**ad**aptive learning method using **ra**ndom feature approximation with multiple **ker**nels, abbreviated as Gradraker. The algorithm takes connection information of a node as input and trains a model to output the corresponding nodal value under supervised learning. Since only vector additions and multiplications are needed, acquiring predictions is convenient and updating the model parameters online becomes possible, which makes the Gradraker algorithm promising to large dynamic networks. What is more, the usage of different kernels or their mixtures might also extend usable applications. Additionally, the Gradraker algorithm reserves nodal privacy to some extent thanks to the incorporation of the random feature approximation [23]. So, the algorithm or its variants are applicable for an extensive set of scenarios, e.g., traffic dynamic estimation, account anomaly detection in social software, recommendation systems, etc. The authors of [33] have shown the impressive performance of the algorithm in terms of Normalized Mean Square Error (NMSE) and its low complexity. Authors of [38] propose a similar algorithm, Graph Kernel Least Mean Squares-Random Fourier Features (GKLMS-RFF), which contains the same model but takes graph-filtered nodal value time sequence of a node as input, instead of the adjacency vector of a node, and provide the convergence condition. Gradraker is extended to exploit multi-hop information for estimation in [38]. We also showed the potential of the Gradraker algorithm to be applied on multi-layer graphs in Section 1.3.

There are few papers guiding how to configure the model in Gradraker-like algorithms, especially in a theoretical view. We aim to fill this gap. The purpose of doing so is not only to have guidance in configuration, but also to have a better understanding on the pros and cons of the algorithm, recognizing its applicable situations and possibly giving hints on the design of its variants. We choose the Single-Kernel Gradraker (SKG) algorithm as the entry point. A Gradraker model consists of several SKG models which share the input of the

Gradraker model. Each component SKG model outputs an estimation which is then used for acquiring the final estimation of the Gradraker model via an aggregation algorithm. That is, SKG models are building blocks of Gradraker-based algorithms and Gradraker performance is highly influenced by the best performance among all the components. Thus, understanding SKG performance in a detailed manner is of great importance for future studies.

To achieve the best performance for an SKG model, there are a few hyperparameters, i.e., the loss function, the learning rate, the number of repeated times for training, the number of random features, and the Gaussian kernel variance, which should be properly chosen. The loss function is selected based on applications. For instance, Least Squares (LS) loss function is usually applied in regression problems. A suitable value of the learning rate is proposed in [39] and we will discuss it in detial in the following chapter. The number of repeated times for training can be found by techniques like monitoring validation loss during training and stopping training when the validation loss does not improve [40] as done in the field of machine learning. The number of random features does not play a major role affecting the model performance once it is sufficiently large. Thus, we will focus on the problem of choosing a suitable kernel for a training set for there is no discussion on it prior to our paper to the best of our knowledge. The study of choosing a suitable kernel is not trivial. Gradraker is proposed using a kernel dictionary with multiple kernels, letting the algorithm choose suitable ones. However, the computational cost grows linearly with the size of the dictionary. What is worse, if the kernel dictionary does not contain any suitable kernel, the performance would be bad, and adding more kernels blindly to the dictionary may not be beneficial. So, distinguishing suitable kernels helps achieve the best performance with the lowest cost. Among many kinds of shift-invariant kernels [23], e.g., Gaussian kernels, Laplacian kernels, and Cauchy kernels, we will focus on Gaussian kernels. Noting that the kernel being used models how similarity changes with difference, and that the laws of large numbers indicate wide application of the Gaussian distribution, it is intuitive to use Gaussian kernels in most situations [33, 41, 38]. For this reason, in this chapter, we will discuss SKG

with a Gaussian kernel in detail.

2.2 Review: Steps When Applying SKG

The basic sequence of steps applying SKG is preparing the training set, building up and initializing the model, training sequentially, and performing prediction (updating trainable parameters if available).

2.2.1 Preparing a Training Set

Let there be a set of sampled nodes $\mathcal{V} = \{v_n\}_{n=1}^{N}$ with known nodal values $\{y_n\}_{n=1}^{N}$, a set of referencing nodes $\mathcal{V}_r = \{v_{r,m}\}_{m=1}^{M}$, and a description of connection between the two sets of nodes. Note that nodal values of referenced nodes do not play a role. As we mentioned earlier, the description of connection can be in the form of a matrix $\mathbf{A}$, of size $M \times N$, whose element $[\mathbf{A}]_{m,n}$ is 0 if the sampled node v_n is not connected with the referencing node $v_{r,m}$, or 1 if the two nodes are connected in the case of unweighted graphs, or the weight over the edge connecting the two nodes in the case of weighted graphs. Note that a column of $\mathbf{A}$ reports the description of connection between the corresponding sampled node and all the referencing nodes. So, we call the description vector *adjacency vector* of the sampled node. Denote the adjacency vector of the sampled node v_n as $\mathbf{a}_n$ with size $M \times 1$. Combining with the nodal value y_n of v_n, we get the pair $(\mathbf{a}_n, y_n)$ for v_n, and the set of pairs $\{(\mathbf{a}_n, y_n)\}_{n=1}^{N}$ is called the training set.

The sampled node set $\mathcal{V}$ and the referencing node set $\mathcal{V}_r$ are not necessarily the same. In [33], $\mathcal{V} = \mathcal{V}_r$, and thus the N adjacency vectors can be formatted in an adjacency matrix of size $N \times N$. In our paper, we generalize applicable scenarios such that $\mathcal{V}_r$ can be any set of nodes, without the need to modify the Gradraker algorithm.

2.2.2 Building Up the Model and Initialization

The model takes an adjacency vector $\mathbf{a}_n$ of $v_n \in \mathcal{V}$ as input. The first part of the SKG model is for acquiring a nonlinear transform $z(\mathbf{a}_n)$ of the input $\mathbf{a}_n$ through a nonlinear mapping $z : \mathbb{R}^{M\times 1} \mapsto \mathbb{R}^{2D\times 1}$. Specifically,

$$\begin{aligned} z(\mathbf{a}_n) =&[\sin(\boldsymbol{\xi}_1^\top \mathbf{a}_n), \sin(\boldsymbol{\xi}_2^\top \mathbf{a}_n), ..., \sin(\boldsymbol{\xi}_D^\top \mathbf{a}_n), \\ &\cos(\boldsymbol{\xi}_1^\top \mathbf{a}_n), \cos(\boldsymbol{\xi}_2^\top \mathbf{a}_n), ..., \cos(\boldsymbol{\xi}_D^\top \mathbf{a}_n)]^\top / \sqrt{D} \end{aligned} \tag{2.1}$$

where $\{\boldsymbol{\xi}_i\}_{i=1}^D$ are random features [23] drawn from a distribution which is the Fourier transform of the kernel κ in SKG. Note we have to manually choose κ. Recall that we will focus on Gaussian kernels in the paper, then the problem is reduced to choosing a variance σ^2 for the Gaussian kernel. Once the kernel is chosen, the following claim is helpful to generate $\{\boldsymbol{\xi}_i\}_{i=1}^D$.

Claim 2.1. *Supposing a Gaussian kernel κ with variance of σ^2, i.e., $\kappa(\boldsymbol{x}_1, \boldsymbol{x}_2) = e^{-\frac{\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|^2}{2\sigma^2}}$, random features $\{\boldsymbol{\xi}_i\}_{i=1}^D$ should be drawn from the Gaussian distribution $\mathcal{N}(0, \sigma^{-2}\boldsymbol{I})$ when using the random feature approximation for κ [23, 33].*

Proof. Since the value of a Gaussian kernel relates with the difference of its input vectors, let us have $\kappa(\mathbf{x}) = e^{-\frac{\|\mathbf{x}\|^2}{2\sigma^2}}$ where $\mathbf{x} = [x_1, ..., x_N]^\top$, and the Fourier transform $\rho_{\mathbf{f}}(\mathbf{f})$ can be written as in [42]

$$\rho_{\mathbf{f}}(\mathbf{f}) = \int_{\mathbf{R}^n} \kappa(\mathbf{x}) e^{-j2\pi \mathbf{f}^\top \mathbf{x}} d\mathbf{x}$$

where $\mathbf{f} = [f_1, f_2, ..., f_N]^\top$, and $\mathbf{f}^\top \mathbf{x}$ denotes the inner product of $\mathbf{f}$ and $\mathbf{x}$, i.e., $\mathbf{f}^\top \mathbf{x} = \sum_{i=1}^N f_i x_i$. Since $\mathbf{x} \in \mathbf{R}^n$, $\|\mathbf{x}\|^2 = \sum_{i=1}^N x_i^2$, then

$$\rho_{\mathbf{f}}(\mathbf{f}) = \int_{-\infty}^{\infty} ... \int_{-\infty}^{\infty} e^{-\frac{\sum_{i=1}^N x_i^2}{2\sigma^2}} e^{-j2\pi \sum_{i=1}^N f_i x_i} dx_1 ... dx_N$$

$$= \prod_{i=1}^{N} \int_{-\infty}^{\infty} e^{-\frac{x_i^2}{2\sigma^2}} e^{-j2\pi f_i x_i} dx_i$$

which can be seen as the product of the Fourier transform for the one-dimensional Gaussian kernels in every dimension. Notice that the product implies independence among dimensions.

It is a well-known Fourier transform pair that

$$e^{-\pi x^2} \overset{\mathcal{F}.\mathcal{T}.}{\longleftrightarrow} e^{-\pi f^2}.$$

Notice the Fourier transform of $g(x)$, $\mathcal{F}\{g(x)\} = G(f) = \int_{-\infty}^{\infty} g(x)e^{-j2\pi fx}dx$. According to the time scaling property of Fourier transform, i.e., $g(ax) \overset{\mathcal{F}.\mathcal{T}.}{\longleftrightarrow} \frac{1}{|a|}G(\frac{f}{a})$ where $g(x) \overset{\mathcal{F}.\mathcal{T}.}{\longleftrightarrow} G(f)$, we have

$$e^{-\frac{x^2}{2\sigma^2}} \overset{\mathcal{F}.\mathcal{T}.}{\longleftrightarrow} \sqrt{2\pi}\sigma e^{-2\pi^2\sigma^2 f^2} = \frac{1}{(\frac{1}{2\pi\sigma})\sqrt{2\pi}} e^{-\frac{f^2}{2(\frac{1}{2\pi\sigma})^2}}.$$

That is, $f \sim \mathcal{N}(0, (\frac{1}{2\pi\sigma})^2)$. For $\xi = 2\pi f$, it is $\xi \sim \mathcal{N}(0, \sigma^{-2})$. Because when X is a random variable and $Y = aX + b$ where a and b are both constants, the probability density function (PDF) of Y is $p_Y(y) = \frac{1}{|a|}p_X(\frac{y-b}{a})$ where $p_X(x)$ denotes the PDF of X.

Similarly, all elements in $\boldsymbol{\xi}$ have Gaussian distribution $\mathcal{N}(0, \sigma^{-2})$, that is, $\boldsymbol{\xi} \sim \mathcal{N}(0, \sigma^{-2}\mathbf{I}_N)$.■

According to **Claim 2.1**, random features $\{\boldsymbol{\xi}_i\}_{i=1}^{D}$ should follow $\mathcal{N}(0, \sigma^{-2}\mathbf{I})$. Note that D is also preselected and that random features are fixed during training and predicting phases once they are chosen.

Then, $z(\mathbf{a}_n)$ goes through the second part of the model, which is linear, and provides a prediction as

$$\hat{f}_n = \boldsymbol{\theta}^\top z(\mathbf{a}_n) \tag{2.2}$$

where $\hat{f}_n$ denotes the prediction. The column vector $\boldsymbol{\theta}$ whose size is $2D \times 1$ is the trainable parameter.

Prior to the training phase, the trainable parameter is initialized as $\boldsymbol{\theta}_0 = \mathbf{0}$. Since $\boldsymbol{\theta}$ is updated every time, we will use $\boldsymbol{\theta}_t$ to denote the $\boldsymbol{\theta}$ value at the end of time (iteration) t.

2.2.3 Sequential Training

The parameter $\boldsymbol{\theta}$ is updated by the gradient descent algorithm, i.e., $\boldsymbol{\theta}_t, 1 \leq t \leq T$ where T denotes the training duration is updated via

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \nabla_{\boldsymbol{\theta}} \mathcal{L}_t \tag{2.3}$$

where $\nabla_{\boldsymbol{\theta}} \mathcal{L}_t$ is the gradient of the loss function $\mathcal{L}$ with respect to $\boldsymbol{\theta}$ at time t and η is the preselected learning rate. Noting that LS loss is used, we have $\mathcal{L}(y_{true}, \hat{f}) = (y_{true} - \hat{f})^2$. Then, the gradient at time t which is employed in (2.3) is

$$\nabla_{\boldsymbol{\theta}} \mathcal{L}_t = -2(y_{n_t} - \hat{f}_{n_t}) z(\mathbf{a}_{n_t}) \tag{2.4}$$

where y_{n_t}, $\hat{f}_{n_t}$, and $\mathbf{a}_{n_t}$ are the true nodal value, the prediction, and the adjacency vector of the node used at time t, respectively. Note that $\mathbf{a}_{n_t}$ is not any specific adjacency vector but random because there is no assumed order for sampled nodes being processed. For notational simplicity, we will use $\mathbf{a}_t$, $\hat{f}_t$, and y_t instead of $\mathbf{a}_{n_t}$, $\hat{f}_{n_t}$, and y_{n_t} from now on.

In [33] and other papers about kernel-based predicting methods, sometimes an overfitting-controlling term is summed with the squared difference $(y_{pred} - y_{true})^2$ in calculating the loss $\mathcal{L}(y_{pred}, y_{true})$. However, the overfitting-controlling term does not greatly affect the level of best performance of SKG achieved on a graph signal, so we ignore it in our analysis.

During the training phase, we process a sampled node at a time, i.e., getting its adjacency vector which is then put through the model to get a prediction, and updating the model. The process is repeated until all sampled nodes are processed. It is possible for the training set to be used multiple times during the training phase, and the number of times the training set is repeatedly used is called the number of epochs, denoted by E. The parameter E is also preselected. The training duration is $T = EN$.

2.2.4 Predicting

When the adjacency vector $\mathbf{a}'$ of a tested node v' to the set of referencing nodes is known, we can use the well-trained model to predict the nodal value. To that end, we first acquire the nonlinear transform $z(\mathbf{a}')$ via (2.1), and then get a prediction via (2.2). If its true value can be known, the trainable parameter $\boldsymbol{\theta}$ of the model can be updated via (2.3).

2.3 Mathematical Tools

A convincing way to illustrate the influence of σ^2 on predictions is to express predictions explicitly in terms of σ^2. It is a hard problem due to the training process. Alternatively, we express predictions as weighted averages of observations, where the influence of σ^2 on the weights is easier to show. To do so, we have to introduce two variables. One is used as weights of observations in predictions, called *contribution weight*. The other one, called *similarity measure*, is an intermediate variable in finding contribution weights. In this section, we give definitions of the two variables, and state their properties; preparing for analysis on how prediction behaves under different σ^2 in the next section. We introduce the *similarity measure* $B_{i,j}$ first as it is basic to the definition of the *contribution weight* $F_{i,j}$.

2.3.1 Similarity Measure

The definition of the similarity measure $B_{i,j}$ for the pair of nodes seen at time i and j, $1 \leq i < j \leq T+1$, is

$$B_{i,j} \triangleq 2\eta z^\top(\mathbf{a}_i) z(\mathbf{a}_j) \tag{2.5}$$

where $\mathbf{a}_i$ and $\mathbf{a}_j$ are the adjacency vectors of nodes used at time i and time j. As its name suggests, $B_{i,j}$ can be seen as a similarity measure between $\mathbf{a}_i$ and $\mathbf{a}_j$ recalling that $\kappa(\mathbf{a}_i, \mathbf{a}_j) \approx z^\top(\mathbf{a}_i) z(\mathbf{a}_j)$ [33], and that a kernel function is a form of similarity measure.

Note that $B_{i,j}$ is a random number. Its randomness comes from both the random features $\{\boldsymbol{\xi}_i\}_{i=1}^D$ and $\mathbf{a}_i$ and $\mathbf{a}_j$ because adjacency vectors vary among different datasets. Even for a given dataset, $\mathbf{a}_i$ and $\mathbf{a}_j$ cannot be determined because of the random sampling for the sampled nodes. Considering the uniform distribution of when a specific node is processed within an epoch, the distribution of $B_{i,j}$ is identical for all qualified pairs of i and j. We keep the indices to denote the time when the adjacency vectors are processed.

Since weights of observations in predictions build on similarity measures $B_{i,j}$, studying properties of $B_{i,j}$ not only helps understanding how σ^2 changes $B_{i,j}$, but also paves a path to the impact of σ^2 on observations weights. So, we illustrate two properties of $B_{i,j}$, *exponential approximation* and *positive average*, in the following.

Substituting (2.1) into (2.5), we have

$$\begin{aligned} B_{i,j} =& \frac{2\eta}{D} \sum_{k=1}^{D} [\sin(\boldsymbol{\xi}_k^\top \mathbf{a}_i) \sin(\boldsymbol{\xi}_k^\top \mathbf{a}_j) + \cos(\boldsymbol{\xi}_k^\top \mathbf{a}_i) \cos(\boldsymbol{\xi}_k^\top \mathbf{a}_j)] \\ =& 2\eta \frac{\sum_{k=1}^{D} \cos[\boldsymbol{\xi}_k^\top (\mathbf{a}_i - \mathbf{a}_j)]}{D} \\ =& 2\eta \frac{\sum_{k=1}^{D} \cos[\boldsymbol{\xi}_k^\top \mathbf{d}_{i,j}]}{D} \end{aligned} \tag{2.6}$$

where $\mathbf{d}_{i,j} = \mathbf{a}_i - \mathbf{a}_j$. It is shown in (2.6) that $B_{i,j}$ is actually the sample average of D terms of $\cos(C_{k,i,j})$ where $C_{k,i,j} = \boldsymbol{\xi}_k^\top \mathbf{d}_{i,j}$ multiplied by a scalar 2η. Whereas, $\mathbf{d}_{i,j}$ is a random vector with respect to different training data and the random processing order of sampled nodes, $\{\boldsymbol{\xi}_k\}_{k=1}^D$ are related to model configuration. We will focus on how $B_{i,j}$ changes with respect to $\{\boldsymbol{\xi}_k\}_{k=1}^D$ (viewing $\mathbf{a}_i$ and $\mathbf{a}_j$ as given for now). Recalling from **Claim 2.1** that elements of $\boldsymbol{\xi}_k, k = 1, ..., D$ are independently and identically distributed (i.i.d.) Gaussian random numbers with variance σ^{-2}, we know that $C_{k,i,j}$ follows $\mathcal{N}(0, \|\mathbf{d}_{i,j}\|^2/\sigma^2)$ for a given $\mathbf{d}_{i,j}$. Notice $\{C_{k,i,j}\}_{k=1}^D$ are i.i.d. because of i.i.d. $\{\boldsymbol{\xi}_k\}_{k=1}^D$. Supposing D is sufficiently large, we can follow the weak law of large numbers and get

$$B_{i,j} \cong 2\eta \mathbb{E}[\cos(C_{k,i,j})|\mathbf{d}_{i,j}]. \tag{2.7}$$

Note $B_{i,j}$ is a random number but varies in a small range given $\mathbf{d}_{i,j}$ and a sufficiently large D. The following claim can be useful to get the explicit expression of the conditional expectation.

Claim 2.2. *Suppose X is a Gaussian random number such that $X \sim \mathcal{N}(0, \sigma_X^2)$. Then, we have*

$$\mathbb{E}[\cos(X)] = e^{-\frac{\sigma_X^2}{2}}, \tag{2.8}$$

and

$$\mathbb{V}[\cos(X)] = \frac{1}{2}(e^{-\sigma_X^2} - 1)^2. \tag{2.9}$$

Proof. The characteristic function [43] of the random variable X is

$$\mathbb{E}[e^{j\upsilon X}] \equiv \psi(j\upsilon) = \int_{-\infty}^{\infty} e^{j\upsilon x} p(x) dx = \mathcal{F}\{p(x)\}$$

where υ is real, $j = \sqrt{-1}$, $\mathcal{F}$ denotes Fourier transform, and $p(x) = \frac{1}{\sigma_X\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_X^2}}$. According to **Claim 2.1**, it is clear that

$$p(x) = \frac{1}{\sigma_X\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma_X^2}} \overset{\mathcal{F}.\mathcal{T}.}{\longleftrightarrow} \psi(j\upsilon) = e^{-\frac{\sigma_X^2\upsilon^2}{2}}.$$

Thus,

$$\mathbb{E}[e^{jX}] = \psi(j\upsilon)|_{\upsilon=1} = e^{-\frac{\sigma_X^2}{2}}.$$

On the other hand, according to Euler's Equation, $e^{jX} = \cos(X) + j\sin(X)$, we have

$$\mathbb{E}[e^{jX}] = \mathbb{E}[\cos(X) + j\sin(X)] = \mathbb{E}[\cos(X)] + j\mathbb{E}[\sin(X)].$$

Since Gaussian PDFs with zero mean is an even function while $\sin(\cdot)$ is odd, i.e., $\mathbb{E}[\sin(X)] = 0$ and

$$\mathbb{E}[\cos(X)] = \mathbb{E}[e^{jX}] = e^{-\frac{\sigma_X^2}{2}}.$$

Similarly,

$$\mathbb{E}[\cos(2X)] = \mathbb{E}[e^{j2X}] = \psi(j\upsilon)|_{\upsilon=2} = e^{-2\sigma_X^2}.$$

Then,

$$\begin{aligned}\mathbb{V}[e^{jX}] =& \mathbb{E}\left[\cos^2(X)\right] - \left(\mathbb{E}\left[\cos(X)\right]\right)^2 = \frac{1}{2}\mathbb{E}\left[\cos(2X)\right] + \frac{1}{2} - \left(\mathbb{E}\left[\cos(X)\right]\right)^2 \\ =& \frac{1}{2}\left(e^{-\sigma_X^2} - 1\right)^2\end{aligned}$$

■

Exponential Approximation

Substituting (2.8) into (2.7), we get the exponential approximation

$$B_{i,j} \cong 2\eta \mathbb{E}[\cos(C_{k,i,j})|\mathbf{d}_{i,j}] = 2\eta e^{-\frac{\|\mathbf{d}_{i,j}\|^2}{2\sigma^2}} \tag{2.10}$$

when D is sufficiently large.

The equality in (2.10) is the same expression as the random feature approximation [23, 33], but in reverse order. Recall that for a Gaussian kernel $\kappa(\mathbf{a}_i, \mathbf{a}_j) = e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2\sigma^2}}$, its mathematical expression of the random feature approximation is $\kappa(\mathbf{a}_i, \mathbf{a}_j) = e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2\sigma^2}} \approx z^\top(\mathbf{a}_i) z(\mathbf{a}_j)$ where $z(\cdot)$ is defined in (2.1). If we replace $B_{i,j}$ and $\mathbf{d}_{i,j}$ in (2.10) with the definition in (2.5) and $\mathbf{a}_i - \mathbf{a}_j$, respectively, we get $2\eta z^\top(\mathbf{a}_i) z(\mathbf{a}_j) \cong 2\eta e^{-\frac{\|\mathbf{a}_i - \mathbf{a}_j\|^2}{2\sigma^2}}$ which is the same as the random feature approximation. The equality in (2.10) explicitly shows how σ^2 affects $B_{i,j}$.

Positive Average

Taking expectation for (2.6), we get

$$\mathbb{E}\left[B_{i,j}\right] = 2\eta \mathbb{E}\left[\mathbb{E}\left[\cos\left(C_{k,i,j}\right)|\mathbf{d}_{i,j}\right]\right]. \tag{2.11}$$

In (2.11), $\mathbb{E}\left[B_{i,j}\right]$ is with respect to the joint distribution of $\{\boldsymbol{\xi}_k\}_{k=1}^D$ and $\mathbf{d}_{i,j}$. On the right hand side, the inner expectation is with respect to the conditional distribution of $\{\boldsymbol{\xi}_k\}_{k=1}^D$ given $\mathbf{d}_{i,j}$ while the outer expectation is with respect to the distribution of $\mathbf{d}_{i,j}$. Recall that when $\mathbf{d}_{i,j}$ is given and D is sufficiently large, $B_{i,j}$ varies within a small range and can be approximated via (2.10). For a real dataset, $\mathbf{d}_{i,j}$ is usually not deterministic but has a

distribution under random sampling without replacement, and thus $B_{i,j}$ may greatly vary with different $\mathbf{d}_{i,j}$ values. From **Claim 2.2**, it is known that $0 < \mathbb{E}\left[\cos\left(C_{k,i,j}\right)|\mathbf{d}_{i,j}\right] \leq 1$ for any $\mathbf{d}_{i,j}$, where the equality holds when $\|\mathbf{d}_{i,j}\|^2 = 0$. So, for real datasets where nonzero $\mathbf{d}_{i,j}$ exists, we get from (2.11) that $0 < \mathbb{E}\left[B_{i,j}\right] < 2\eta$.

2.3.2 Contribution Weight

As we mentioned before, we aim to express a prediction of SKG in terms of observations. We are able to do so via contribution weights introduced in the following. The definition of the contribution weights $F_{i,j}$ for the pair of nodes seen at time i and j, $1 \leq i < j \leq T+1$ is

$$F_{i,j} \triangleq \begin{cases} B_{i,j}, & \text{for } i = j-1, \\ B_{i,j} - \sum_{k=i+1}^{j-1} B_{i,k} F_{k,j}, & \text{for } 1 \leq i < j-1, \end{cases} \tag{2.12}$$

and undefined otherwise. Because of the randomness in $B_{p,q}, i \leq p \leq q \leq j$, $F_{i,j}$ is also a random number. The definition in (2.12) indicates that $F_{i,j}$ is affected by σ^2 indirectly via $B_{i,j}$.

We show two useful properties for $F_{i,j}$, *weighting* and *conformity with* $B_{i,j}$.

Weighting Property

Firstly, the following claim shows $\{F_{i,j}\}_{i=1}^{j-1}$ are used as coefficients of previously seen nodal values in prediction.

Claim 2.3. *Assume we are applying SKG on a training set. During the training phase, at*

time $t, 1 < t \leq T$, *we have*

$$\hat{f}_t = \sum_{i=1}^{t-1} y_i F_{i,t} \tag{2.13}$$

where $\hat{f}_t$ *denotes the prediction at time* t, y_i *denotes the true nodal value at training time* i, *and* $\{F_{i,t}\}_{i=1}^{t-1}$ *are defined as in (2.12).*

Proof. First we prove for the trainable parameter $\boldsymbol{\theta}$ at time t that

$$\boldsymbol{\theta}_t = 2\eta \sum_{i=1}^{t} e_i z(\mathbf{a}_i)$$

where e_i is the predicting error at time i, $e_i = y_i - \hat{f}_i$. Note that the square loss is $\mathcal{L}(y_{pred}, y_{true}) = (y_{true} - y_{pred})^2$ in our case, the loss at time t is

$$\mathcal{L}(y_t, \boldsymbol{\theta}_{t-1}^\top z(\mathbf{a}_t)) = (f_t - \boldsymbol{\theta}_{t-1}^\top z(\mathbf{a}_t))^2.$$

Then the gradient of the loss function with respect to $\boldsymbol{\theta}_{t-1}$, i.e., g_t, is

$$g_t = \nabla_{\boldsymbol{\theta}_{t-1}} \mathcal{L}(y_t, \boldsymbol{\theta}_{t-1}^\top z(\mathbf{a}_t)) = -2(f_t - \boldsymbol{\theta}_{t-1}^\top z(\mathbf{a}_t)) z(\mathbf{a}_t) = -2e_t z(\mathbf{a}_t)$$

which uses

$$\nabla_{\mathbf{x}}(\mathbf{x}^\top \mathbf{a}) = \nabla_{\mathbf{x}}(\mathbf{a}^\top \mathbf{x}) = [\frac{\partial(\mathbf{a}^\top \mathbf{x})}{\partial x_1}, \frac{\partial(\mathbf{a}^\top \mathbf{x})}{\partial x_2}, ..., \frac{\partial(\mathbf{a}^\top \mathbf{x})}{\partial x_N}]^\top = [a_1, a_2, ..., a_N]^\top = \mathbf{a}$$

where $\mathbf{a} = [a_1, a_2, ..., a_N]^\top$ and $\mathbf{x} = [x_1, x_2, ..., x_N]^\top$. Denote η as the learning rate, then

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta g_t = \boldsymbol{\theta}_{t-1} + 2\eta e_t z(\mathbf{a}_t).$$

Tracing back to $\boldsymbol{\theta}_0$, $\boldsymbol{\theta}_t$ can be rewritten in terms of $\boldsymbol{\theta}_0$ as

$$\begin{aligned}\boldsymbol{\theta}_t &= \boldsymbol{\theta}_{t-1} + 2\eta e_t z(\mathbf{a}_t) = \boldsymbol{\theta}_{t-2} + 2\eta e_{t-1} z(\mathbf{a}_{t-1}) + 2\eta e_t z(\mathbf{a}_t) \\ &= \boldsymbol{\theta}_0 + 2\eta \sum_{i=1}^{t} [e_i z(\mathbf{a}_i)].\end{aligned}$$

Together with the initialization $\boldsymbol{\theta}_0 = \mathbf{0}$, we get

$$\boldsymbol{\theta}_t = 2\eta \sum_{i=1}^{t} [e_i z(\mathbf{a}_i)].$$

Then the prediction at time t, $\hat{f}_t$, can be expressed as

$$\hat{f}_t = \boldsymbol{\theta}_{t-1}^\top z(\mathbf{a}_t) = 2\eta \sum_{i=1}^{t-1} [e_i z^\top(\mathbf{a}_i) z(\mathbf{a}_t)] = \sum_{i=1}^{t-1} e_i B_{i,t}$$

which uses the definition of $B_{i,j}$ in (2.5). Now we are prepared to prove the claimed equality. First, check the claimed equality for $t = 2, 3, 4$. It should be

$$\begin{aligned}\hat{f}_2 =& y_1 B_{1,2} = y_1 F_{1,2} \\ \hat{f}_3 =& y_2 B_{2,3} + y_1 (B_{1,3} - B_{1,2} B_{2,3}) = y_2 F_{2,3} + y_1 F_{1,3} \\ \hat{f}_4 =& y_3 B_{3,4} + y_2 (B_{2,4} - B_{2,3} B_{2,4}) + y_1 [B_{1,4} - B_{1,3} B_{3,4} - B_{1,2} (B_{2,4} - B_{2,3} B_{3,4})] \\ =& y_3 F_{3,4} + y_2 F_{2,4} + y_1 F_{1,4}\end{aligned}$$

from which the equality holds. Suppose there exists $t \geq 4$ such that

$$\hat{f}_k = \sum_{i=1}^{k-1} y_i F_{i,k}$$

where

$$F_{i,k} = \begin{cases} B_{i,k}, & \text{if } i = k-1, \\ B_{i,k} - \sum_{j=i+1}^{k-1} B_{i,j} F_{j,k}, & \text{if } 1 \le i < k-1. \end{cases}$$

holds for all integers $k \in \{2, 3, ..., t\}$. Then,

$$\hat{f}_{t+1} = \sum_{i=1}^{t} e_i B_{i,t+1} = \sum_{i=1}^{t} (y_i - \hat{f}_i) B_{i,t+1}.$$

For $\hat{f}_1 = 0$, we have

$$\begin{aligned} \hat{f}_{t+1} &= \sum_{i=1}^{t} y_i B_{i,t+1} - \sum_{i=2}^{t} \hat{f}_i B_{i,t+1} \\ &= \sum_{i=1}^{t} y_i B_{i,t+1} - \sum_{i=2}^{t} \sum_{j=1}^{i-1} y_j F_{j,i} B_{i,t+1} = \sum_{i=1}^{t} y_i B_{i,t+1} - \sum_{j=1}^{t-1} \sum_{i=j+1}^{t} y_j F_{j,i} B_{i,t+1} \\ &= y_t B_{t,t+1} + \sum_{i=1}^{t-1} y_i (B_{i,t+1} - \sum_{j=i+1}^{t} F_{i,j} B_{j,t+1}) \\ &= y_t B_{t,t+1} + \sum_{i=1}^{t-1} y_i \tilde{F}_{i,t+1} \end{aligned}$$

which shows the coefficient of y_t here is $B_{t,t+1}$. Since the coefficient of y_t in calculating $\hat{f}_{t+1}$ is $F_{t,t+1}$ according to definition of $F_{i,j}$ in (2.12), the last equality justifies $F_{t,t+1} = B_{t,t+1}$. Besides, the coefficient for y_{t-1} is

$$\tilde{F}_{t-1,t+1} = B_{t-1,t+1} - F_{t-1,t} B_{t,t+1} = B_{t-1,t+1} - B_{t-1,t} F_{t,t+1} = F_{t-1,t+1}.$$

Coefficients for $y_i, i = 1, ..., t-2$ are

$$\tilde{F}_{i,t+1} = B_{i,t+1} - \sum_{j=i+1}^{t} F_{i,j} B_{j,t+1}$$

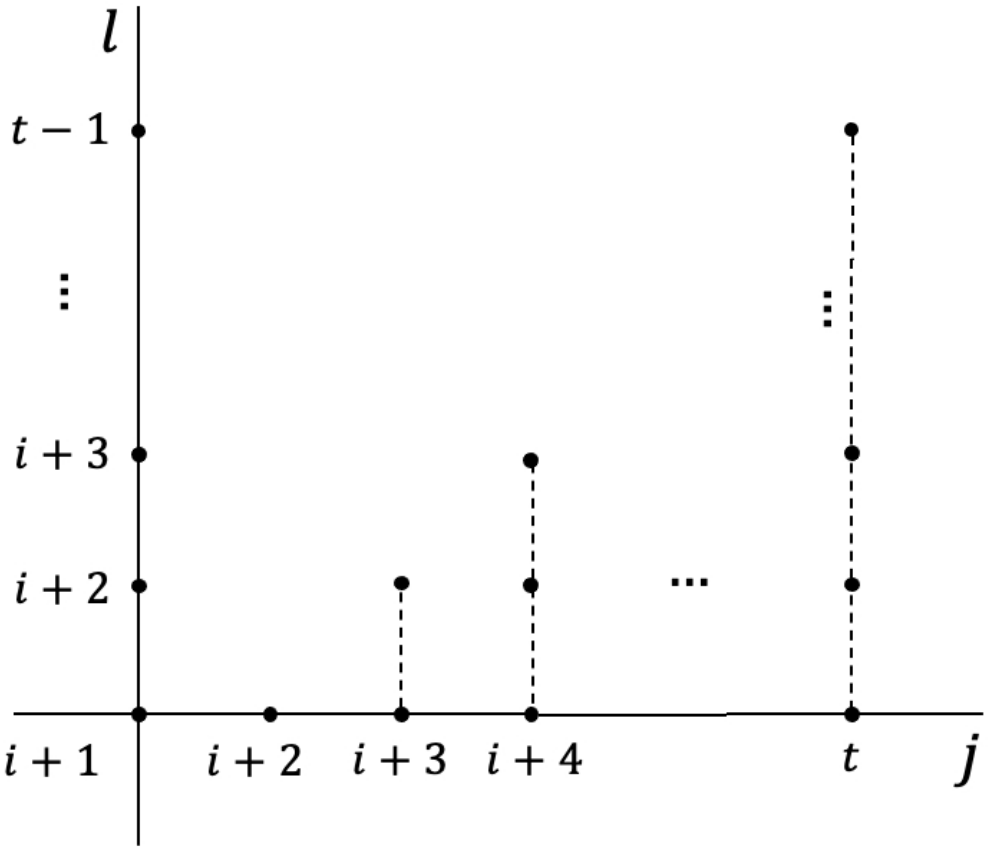

Figure 2.1: Index variables used for the double-summation term in the Proof of **Claim 2.3** .

$$
\begin{aligned}
&= B_{i,t+1} - [B_{i,i+1}B_{i+1,t+1} + \sum_{j=i+2}^{t}(B_{i,j} - \sum_{l=i+1}^{j-1} B_{i,l}F_{l,j})B_{j,t+1}] \\
&= B_{i,t+1} - (B_{i,i+1}B_{i+1,t+1} + \sum_{j=i+2}^{t} B_{i,j}B_{j,t+1} - \sum_{j=i+2}^{t}\sum_{l=i+1}^{j-1} B_{i,l}F_{l,j}B_{j,t+1}).
\end{aligned}
$$

With the order change of indices for the double summation term in the last equation, (see

Fig. 2.1 for reference) we will get

$$\begin{aligned}\tilde{F}_{i,t+1} =& B_{i,t+1} - (\sum_{j=i+1}^{t-1} B_{i,j}B_{j,t+1} + B_{i,t}B_{t,t+1} - \sum_{l=i+1}^{t-1}\sum_{j=l+1}^{t} B_{i,l}F_{l,j}B_{j,t+1}) \\ =& B_{i,t+1} - [B_{i,t}B_{t,t+1} + \sum_{j=i+1}^{t-1} B_{i,j}(B_{j,t+1} - \sum_{l=j+1}^{t} F_{j,l}B_{l,t+1})] \\ =& B_{i,t+1} - (B_{i,t}F_{t,t+1} + \sum_{j=i+1}^{t-1} B_{i,j}F_{j,t+1}) = F_{i,t+1}\end{aligned}$$

where

$$F_{i,t+1} = \begin{cases} B_{i,t+1}, & \text{for } i = t, \\ B_{i,t+1} - \sum_{j=i+1}^{t} B_{i,j}F_{j,t+1}, & \text{for } 1 \leq i < t. \end{cases}$$

Note that we are considering the training phase, t is upper-bounded by T where T denotes the training duration. ■

Although **Claim 2.3** mentions the training phase only, it is easy to extend (2.13) to the predicting (testing) phase. If the nodal value for a tested node is known somehow, the node will work as a sampled node and the SKG model can be trained further in which case (2.13) works fine for the tested node directly. If a tested node comes without a true nodal value, it will have no impact on the model, and all such nodes share exactly the same model in which case all these nodes can be seen as the node at time $T + 1$. In practice, tested nodes with and without known nodal values may be mixed, however, only those with known nodal values will affect the model and later predictions. So, without loss of generality, we will only consider the case where nodal values for tested nodes are unknown. Then, the prediction for

a tested node can be expressed as

$$\hat{f}_{T+1} = \sum_{i=1}^{T} y_i F_{i,T+1} \tag{2.14}$$

where $\hat{f}_{T+1}$ represents the prediction of the tested node. Note that at the time, we only show a prediction is a weighted summation of observations. Based on the following claim, we could step further and show a prediction can be a weighted average of observations using $F_{i,T+1}$.

Claim 2.4. *According to the definition in (2.12), we can get the expectation of the sum of $F_{i,T+1}$ with all qualified i, i.e., $1 \leq i \leq T$, as*

$$\mathbb{E}\left[\sum_{i=1}^{T} F_{i,T+1}\right] = 1 - (1-b)^T$$

where $b = \mathbb{E}[B_{i,j}]$ and $1 \leq i < j \leq T+1$.

Proof. Using $F_{i,j}$'s definition, its expectation is

$$\mathbb{E}[F_{i,j}] = \mathbb{E}[B_{i,j} - \sum_{k=1}^{j-i-1} B_{i,j-k} F_{j-k,j}] = \mathbb{E}[B_{i,j}] - \sum_{k=1}^{j-i-1} \mathbb{E}[B_{i,j-k} F_{j-k,j}]$$

when $1 \leq i < j \leq T+1$ where T is the training duration. Note that $B_{i,j-k}$ and $F_{j-k,j}$ are uncorrelated when $1 \leq k \leq j-i-1$ and the distribution of $B_{i,j}, 1 \leq i < j \leq T+1$ is independent of its indices i and j when $\mathbf{a}_n$ is considered following the same distribution for $1 \leq n \leq T+1$. Denote $\mathbb{E}[B_{i,j}] = b$, we get

$$\mathbb{E}[F_{i,j}] = b\left(1 - \sum_{k=1}^{j-i-1} \mathbb{E}[F_{j-k,j}]\right).$$

It can be verified that

$$\mathbb{E}[F_{j-1,j}] = \mathbb{E}[B_{j-1,j}] = b \tag{2.15}$$

which justifies the claimed equality for $i = j-1, 1 < j \leq T+1$. Suppose there exists $1 \leq t \leq T-1$ such that the claimed equality holds for $i = j-l, l < j \leq T+1$ when $1 \leq l \leq t$, then for $i = j-(t+1), t+1 < j \leq T+1$ we have

$$\mathbb{E}[F_{j-t-1,j}] = b\left(1 - \sum_{k=1}^{t} \mathbb{E}[F_{j-k,j}]\right) = b\left(1 - \frac{b[1-(1-b)^t]}{1-(1-b)}\right) = b(1-b)^t$$

Then, substituting $t = T$, we get the expectation of the sum of $F_{i,T+1}$ with all qualified i, i.e., $1 \leq i \leq T$, as

$$\mathbb{E}\left[\sum_{i=1}^{T} F_{i,T+1}\right] = \sum_{i=1}^{T} \mathbb{E}[F_{i,T+1}] = \sum_{i=1}^{T} b(1-b)^{T-i} = 1-(1-b)^T.$$

■

It has been confirmed by the positive average property of $B_{i,j}$ in Section 2.3.1 that $0 < b < 2\eta$. For most cases where $\eta \ll 1$, we get

$$\lim_{T\to\infty} \mathbb{E}\left[\sum_{i=1}^{T} F_{i,T+1}\right] = 1. \tag{2.16}$$

In practice, the training duration T is usually more than hundreds which is sufficient to get $\mathbb{E}\left[\sum_{i=1}^{T} F_{i,T+1}\right] \approx 1$. The small variance of the summation is observed from experiments such that the summation is close to 1. Thus, using (2.16) together with **Claim 2.3**, we can draw a conclusion that, when T is suitable, the SKG prediction for a tested node is actually a weighted average of all previously seen nodal values, and how much contribution that the nodal value seen at time i makes to the prediction is determined by the Contribution Weight

$F_{i,T+1}$. We call this the weighting property of $F_{i,T+1}$.

Conformity Between $B_{i,j}$ and $F_{i,j}$

It can be observed from simulations that although $F_{i,j}$ for any $1 < j \leq T+1$ increases about exponentially when $1 \leq i < j$, $F_{i,j}$ tends to be greater when $B_{i,j}$ is obviously larger than $\mathbb{E}[B_{i,j}]$. Due to the recursive definition of $F_{i,j}$, the direct derivation between $B_{i,j}$ and $F_{i,j}$ becomes complex. So, we explain the conformity between $B_{i,j}$ and $F_{i,j}$ using induction.

For $j > 1$, because $F_{j-1,j} = B_{j-1,j}$, there is no question that $B_{j-1,j}$ and $F_{j-1,j}$ will be both large or small.

For $j > 2$, we see

$$\begin{aligned} F_{j-2,j} &= B_{j-2,j} - B_{j-2,j-1}F_{j-1,j} = B_{j-2,j} - B_{j-2,j-1}B_{j-1,j} \\ &= B_{j-2,j}\left(1 - \frac{B_{j-2.j-1}B_{j-1,j}}{B_{j-2,j}}\right). \end{aligned}$$

Assuming a sufficiently large D, we consider the exponential approximation of $B_{i,j}$ and get

$$\frac{B_{j-2,j-1}B_{j-1,j}}{B_{j-2,j}} \approx \frac{2\eta e^{-\frac{\|\mathbf{d}_{j-2,j-1}\|^2}{2\sigma^2}} \cdot 2\eta e^{-\frac{\|\mathbf{d}_{j-1,j}\|^2}{2\sigma^2}}}{2\eta e^{-\frac{\|\mathbf{d}_{j-2,j}\|^2}{2\sigma^2}}} = 2\eta e^{-\frac{\|\mathbf{d}_{j-2,j-1}\|^2 + \|\mathbf{d}_{j-1,j}\|^2 - \|\mathbf{d}_{j-2,j}\|^2}{2\sigma^2}}.$$

From Triangle Inequality, we know that $\|\mathbf{d}_{j-2,j-1}\|^2 + \|\mathbf{d}_{j-1,j}\|^2 - \|\mathbf{d}_{j-2,j}\|^2 \geq 0$, so

$$0 < \frac{B_{j-2,j-1}B_{j-1,j}}{B_{j-2,j}} \leq 2\eta.$$

Set $\alpha_2 = \frac{B_{j-2,j-1}B_{j-1,j}}{B_{j-2,j}}$, we can express $F_{j-2,j}$ as

$$F_{j-2,j} = B_{j-2,j}(1 - \alpha_2) \tag{2.17}$$

where $0 < \alpha_2 \ll 1$ since the learning rate η is usually a small number. Noting that α_2 depends on the difference between j and $j-2$ which is 2 but not on j, we denote the fraction to be α_2. From (2.17), we see that $F_{j-2,j}$ and $B_{j-2,j}$ would also be both large or small although there exists the factor $1-\alpha_2$.

For i, j satisfying $i = j-3, j > 3$, we get

$$F_{j-3,j} = B_{j-3,j}[1 - \alpha_{3,2}(1-\alpha_2) - \alpha_{3,1}] \tag{2.18}$$

where $\alpha_{3,2} = \frac{B_{j-3,j-2}B_{j-2,j}}{B_{j-3,j}}$ and $\alpha_{3,1} = \frac{B_{j-3,j-1}B_{j-1,j}}{B_{j-3,j}}$, and thus $0 < \alpha_{3,1}, \alpha_{3,2} \ll 1$. For $\alpha_2, \alpha_{3,1}$ and $\alpha_{3,2}$ all being small positive numbers, it is reasonable to have the following

$$1 - \alpha_{3,2}(1-\alpha_2) - \alpha_{3,1} = 1 - \alpha_{3,1} - \alpha_{3,2}(1-\alpha_2) \approx (1-\alpha_{3,2})(1-\alpha_2). \tag{2.19}$$

The approximation is valid because $\alpha_{3,1}$ and α_2 are small positive numbers. It would be an equality instead when $\alpha_{3,1} = \alpha_2$. Similarly, $(1-\alpha_{3,2})(1-\alpha_2)$ can be considered as a square term because $\alpha_{3,2}$ and α_2 are small positive numbers. In other words, we could choose α_3 to satisfy

$$1 - \alpha_{3,2}(1-\alpha_2) - \alpha_{3,1} = (1-\alpha_3)^2 \tag{2.20}$$

and α_3 is close to $\alpha_2, \alpha_{3,1}$ and $\alpha_{3,2}$. For example, in an experiment with $\eta = 0.05$, it is possible to see $\alpha_2 = 0.08$, $\alpha_{3,1} = 0.06$, and $\alpha_{3,2} = 0.1$. Then, we should choose $\alpha_3 = 0.079$ which is around $\alpha_2, \alpha_{3,1}$ and $\alpha_{3,2}$ to satisfy (2.20). Substituting (2.20) back to (2.18), we would get

$$F_{j-3,j} = B_{j-3,j}(1-\alpha_3)^2 \tag{2.21}$$

which implies that $B_{j-3,j}$ and $F_{j-3,j}$ are related with a factor $(1-\alpha_3)^2$.

Using mathematical induction we get the mathematical expression for the conformity property between $B_{i,j}$ and $F_{i,j}$ for $1 \leq i < j \leq T+1$,

$$F_{i,j} = B_{i,j}(1-\alpha_{j-i})^{j-i-1} \tag{2.22}$$

where α_{j-i} is a small positive number.

The conformity property between $B_{i,j}$ and $F_{i,j}$ considers an exponential term, implying that when $j-i$ is small, it is easier to observe $F_{i,j}$ and $B_{i,j}$ to be large or small at the same time. However, when $j-i$ is large, one observes small $F_{i,j}$ values no matter what $B_{i,j}$ is.

2.4 Gaussian Variance for a Graph

2.4.1 Impact of σ^2 on Predictions

We illustrate how prediction changes when σ^2 increases. Based on different behavior of $B_{i,T+1}$, $F_{i,T+1}$, we divide the possible range of σ^2, $(0,+\infty)$, into four parts, i.e., the Chaos Range, the Extending Range, the Disturbing Range, and the Averaging Range, as shown in Fig. 2.2. We can see from the figure that the applicable interval of σ^2 is divided into four parts, i.e., the chaos range, the extending range, the disturbing range, and the averaging range, from left to right. The boundaries are denoted by σ^2_{ce}, σ^2_{ed}, and σ^2_{ea}, respectively.

Chaos Range

In this range, σ^2 is so small that $B_{i,T+1}$ is close to 0 when $\mathbf{a}_i \neq \mathbf{a}_{T+1}$.

Following the conformity property, $F_{i,T+1}$ keeps pace with $B_{i,j}$. Note that in this case, α_{T+1-i} values are generally close to 0. Take $\alpha_2 \approx 2\eta e^{-\frac{\|\mathbf{d}_{j-2,j-1}\|^2+\|\mathbf{d}_{j-1,j}\|^2-\|\mathbf{d}_{j-2,j}\|^2}{2\sigma^2}}$ as an example.

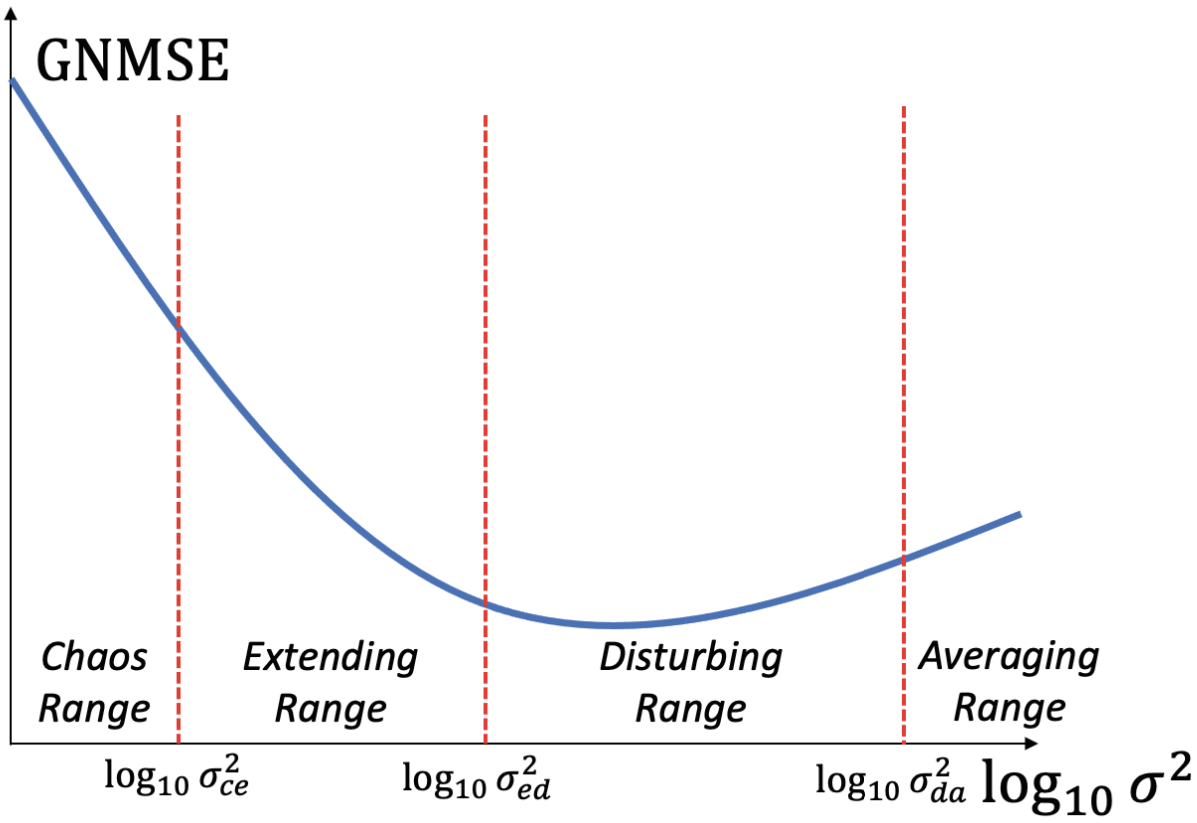

Figure 2.2: A typical GNMSE curve with respect to σ^2.

Because of the small σ^2 value, the exponent is a large negative number when the numerator of the exponent is nonzero. The small α_{T+1-i} values make the exponential term in (2.22) decay slowly with decreasing i from T, resulting in $F_{i,T+1}$ following $B_{i,T+1}$ closely. Like $B_{i,T+1}$, $F_{i,T+1}$ takes positive or negative values. Whereas positive $F_{i,T+1}$ values are viewed as weights of previously-seen nodal values contributing to the prediction, negative $F_{i,T+1}$ values play a disturbing role. Specifically, negative $F_{i,T+1}$ values cancel out positive $F_{i,T+1}$ with similar absolute values, resulting in taking nodal value difference instead of nodal values into consideration for predicting. Thus, we can find a minimum range which negative $F_{i,T+1}$ values fall in, and together with its positive counterpart, we get a symmetric range around 0 which we call the *noise range*. When $F_{i,T+1}$ takes value in the noise range, we say the corresponding nodes acquire an insignificant weight and do not contribute to the prediction.

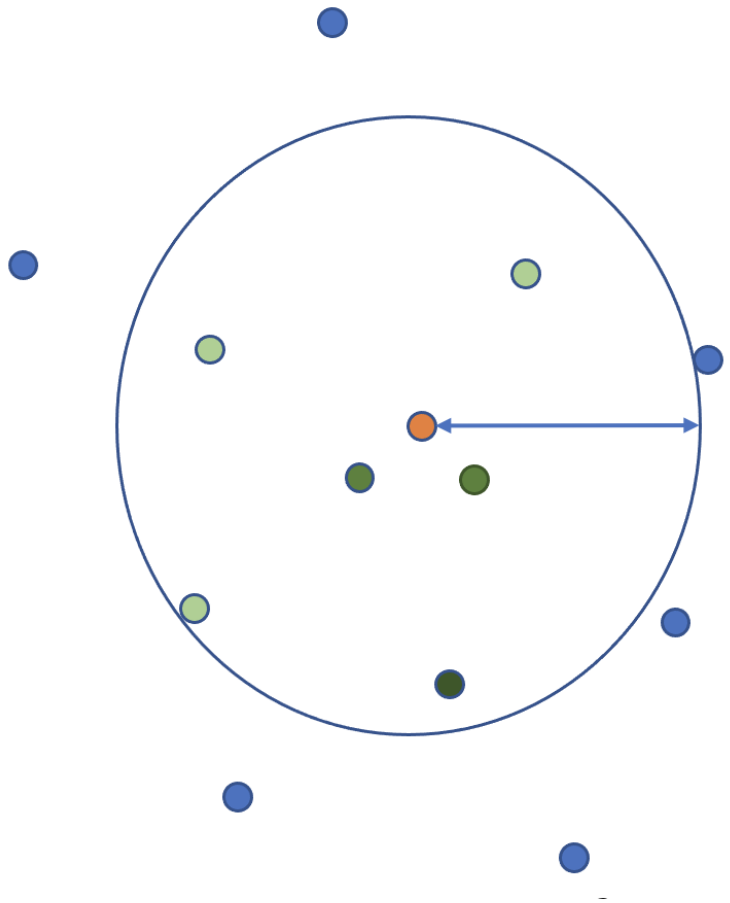

Figure 2.3: A simplified spacial illustration of sampled nodes and a tested node based on $\|\mathbf{d}_{i,T+1}\|^2$.

Since most of $F_{i,T+1}$ values fall into the noise range when σ^2 is in the Chaos Range, the output is less predictable.

Extending Range

When σ^2 is in this range, $B_{i,T+1}$ values with small $\|\mathbf{d}_{i,T+1}\|^2$ are significantly greater than 0 whereas $B_{i,T+1}$ values with large $\|\mathbf{d}_{i,T+1}\|^2$ are still close to 0.

Considering the conformity between $B_{i,j}$ and $F_{i,j}$, it is expected that $F_{i,T+1}$ values for those small $\|\mathbf{d}_{i,T+1}\|^2$ are significantly larger than 0 while $F_{i,T+1}$ values for large $\|\mathbf{d}_{i,T+1}\|^2$ are close to 0. Due to the variation in $B_{i,T+1}$, the noise range still exists. However, there are $F_{i,T+1}$

values falling out of the noise range. Let us call $B_{i,T+1}$ whose corresponding $F_{i,T+1}$ falls outside the noise range the *chosen* $B_{i,T+1}$. With the exponential approximation of $B_{i,j}$, we can find corresponding $\|\mathbf{d}_{i,T+1}\|^2$ for chosen $B_{i,T+1}$ and have the maximum as the *efficient distance*. Fig. 2.3 shows a simplified spatial distribution for sampled nodes and a tested node. In the figure, the orange node represents the tested node and the others are sampled nodes. A circle is centered at the tested node with the radius equal to the efficient distance. The different lightness of green nodes implies different weights. With the efficient distance as the radius, the circle centered at the tested node divides the sampled nodes into two groups. Then, we rewrite the prediction as

$$\hat{f}_{T+1} = \sum_{F_{i,T+1} \text{ is significant}} y_i F_{i,T+1} + \sum_{F_{i,T+1} \text{ is insignificant}} y_i F_{i,T+1}. \tag{2.23}$$

Sampled nodes inside the circle contribute to the first summation in (2.23) with chosen $B_{i,T+1}$ and significant $F_{i,T+1}$. Sampled nodes outside the circle are assigned insignificant weights that fall into the noise range, so their nodal values contribute to the second summation which is less predictable. Clearly, if the sampled nodes inside the circle have nodal values close to that of the tested node, and if the second summation in (2.23) is not dominant, the prediction would be close to its ground truth.

When σ^2 is increased within the Extending Range, the efficient distance grows. That is, the circle is extending to include more sampled nodes. This brings two benefits. First, more nodes are taken into consideration, instead of just a few nodes. Note that sampled nodes inside the circle have significant influence on prediction. When the circle includes only a few nodes all of which happen to have dissimilar nodal values with the tested node, the prediction would not be ideal. Enlarging the circle by increasing σ^2 within the Extending Range is helpful to include more sampled nodes, lowering weights of nodes with dissimilar nodal values. Second, fewer nodes take part into the less predictable part when the efficient distance grows. As a result, the performance of the SKG model gets better as σ^2 increases

within the Extending Range.

Disturbing Range

In this range, $B_{i,T+1}$ for all $\|\mathbf{d}_{i,T+1}\|^2$ becomes significantly larger than 0 but $B_{i,T+1}$ with small $\|\mathbf{d}_{i,T+1}\|^2$ are significantly greater than $B_{i,T+1}$ with large $\|\mathbf{d}_{i,T+1}\|^2$. Notice that it is the relative value not the absolute value of $B_{i,T+1}$ that carries information of similarity in adjacency vectors.

We can calculate an efficient distance using chosen $B_{i,T+1}$ values and draw a circle as in Fig. 2.3. However, the circle loses its role as a boundary. In fact, the circle includes most, if not all, of sampled nodes. Not all nodes inside the circle are assigned significant weights. Sampled nodes with higher $B_{i,T+1}$ still tend to get higher weights, but other sampled nodes would get significant weights if they show up at later times. Take the last sampled node in training as an example. It is assigned weight $F_{T,T+1} = B_{T,T+1}$ which is significantly greater than 0. That is, the last nodal value is considered in prediction regardless of whether the node is spatially close to the tested node or not. In other words, predictions consider closeness in time in addition to similarity among adjacent vectors.

When σ^2 is increased within the Disturbing Range, $B_{i,T+1}$ values generally grow and predictions are focusing more and more on proximity of time. If the recent nodes do not happen to have similar nodal values with the tested node, the prediction will be far from its ground truth.

Averaging Range

In this range, σ^2 is so large that $B_{i,T+1}$ are close to 2η for all $\|\mathbf{d}_{i,T+1}\|^2$.

Note that α_{T+1-i} values are close to 2η in this case. For example, if we look at α_2 we have

$\alpha_2 \approx 2\eta e^{-\frac{\|\mathbf{d}_{j-2,j-1}\|^2+\|\mathbf{d}_{j-1,j}\|^2-\|\mathbf{d}_{j-2,j}\|^2}{2\sigma^2}} \approx 2\eta$. Consequently, $F_{i,T+1}$ is close to an exponential function with the base $1-2\eta$ as i goes from 1 to T.

Because of the exponential shape of $F_{i,T+1}$ with respect to i, it is unsurprising that only recent nodes are taken into consideration in predicting. Besides, it is the same set of sampled nodes that take significant weights in calculation for different tested nodes, and $\{F_{i,T+1}\}_{i=1}^{T}$ are similar for different tested nodes. Thus, it is anticipated that outputs of the model are about the same for all tested nodes.

2.4.2 How to Choose a σ^2

For clarity, let us denote the boundary between the Chaos Range and the Extending Range by σ_{ce}^2, the boundary between the Extending Range and the Disturbing Range by σ_{ed}^2, and the boundary between the Disturbing Range and the Averaging Range by σ_{da}^2. From the analysis in Section 2.4.1, we conclude that the performance is bad in the Chaos Range, gets better in the Extending Range, might get worse in the Disturbing Range, and is bad in the Averaging Range. As predictions in the Disturbing Range consider more proximity of time instead of network topology than in the Extending Range, we choose the boundary between the Extending Range and the Disturbing Range σ_{ed}^2 as a suitable σ^2 (cases where performance gets the best in the Disturbing Range are discussed in Section 2.6.2). Using Fig. 2.3 as an illustration, the radius of the circle achieves its maximum while not including nodes with dissimilar adjacency vectors on this boundary. Intuitively speaking, what happens at σ_{ed}^2 is $B_{i,T+1}$ values for large $\|\mathbf{d}_{i,T+1}\|^2$ are "just significantly greater than 0."

We should find the largest possible $\|\mathbf{d}_{i,T+1}\|^2$ value. However, we cannot know the distribution of $\|\mathbf{d}_{i,T+1}\|^2$ when we configure the SKG model. So, we use the largest value of $\|\mathbf{d}_{i,j}\|^2$ among all pairs of sampled nodes, denoted by $\|\mathbf{d}\|_{max}^2$, instead. We also need to give a concrete math expression for "significantly greater than 0." Recall that $F_{i,T+1}$ is considered as

significant if it falls out of the noise range. Following the conformity property between $B_{i,j}$ and $F_{i,j}$, $F_{i,T+1}$ is likely to fall out of the noise range when $B_{i,T+1}$ falls out of the noise range. Then, $B_{i,T+1}$ is significant when it is greater than the upper bound of the noise range.

It can be observed that the noise range exists for different σ^2 values. The existence is (partly) due to the variation of $B_{i,T+1}$ for $1 \leq i \leq T$. For example, in an extreme case where $B_{i,T+1} = 2\eta$ for $1 \leq i \leq T$, $F_{i,T+1}$ is an exact exponential function with respect to i and the noise range vanishes. Detailed analysis on the noise range is left for future study. Although the noise range changes with $B_{i,T+1}$ variation, the change is limited. So, we can calculate the noise range in the Chaos Range which is easier to derive and use it for all σ^2. Recall in the Chaos Range, $F_{i,T+1}$ values closely follow corresponding $B_{i,T+1}$ values. From (2.9), it is known that

$$\mathbb{V}[B_{i,j}|\mathbf{d}_{i,j}] = \frac{(2\eta)^2}{2D}\Big(e^{-\frac{\|\mathbf{d}_{i,j}\|^2}{\sigma^2}} - 1\Big)^2$$

which is a decreasing function with respect to $\|\mathbf{d}_{i,j}\|^2$. That is, $\mathbb{V}[B_{i,T+1}]$ which is an upper bound of $\mathbb{V}[F_{i,T+1}]$ is upper-bounded by $\mathbb{V}[B_{i,T+1}|\mathbf{0}] \leq \frac{(2\eta)^2}{2D}$. Then, the upper bound of the noise range $noise_{up}$ can be approximated by the standard deviation as

$$noise_{up} \approx \sqrt{\frac{(2\eta)^2}{2D}} = \frac{1}{\sqrt{2D}} \times 2\eta. \tag{2.24}$$

For a more precise noise range at the boundary between the Extending Range and the Disturbing Range, please follow Algorithm 3.

Once making sure $noise_{up}$ and $\|\mathbf{d}\|^2_{max}$, we can apply the exponential approximation of $B_{i,T+1}$ and get

$$2\eta e^{-\frac{\|\mathbf{d}\|^2_{max}}{2\sigma^2_{ed}}} = noise_{up}$$

which is equivalent to

$$\sigma_{ed}^2 = -\frac{\|\mathbf{d}\|_{max}^2}{2\ln\frac{noise_{up}}{2\eta}}. \tag{2.25}$$

The steps of how to choose a suitable σ^2 are summarized in Algorithm 2. We would like to mention that, although (2.24) and (2.25) indicate that D affects calculated σ_{ed}^2, the best σ^2 is not influenced by D theoretically as long as D is sufficiently large. Note that D cannot be arbitrarily small for the validity of the random feature approximation. The proposed σ^2 is close to the optimal one, but not exactly the same. In this case, (2.24) provides an approximation of $noise_{up}$ and we also provide Algorithm 3 to mitigate the impact of D on the proposed σ^2.

Algorithm 2 Choosing σ^2 for the Gaussian Kernel in SKG

Input: adjacency vectors for all sampled nodes, and the number of the random features D.

Get $noise_{up}$ via (2.24) (alternatively, for a more precise noise range, use **Algorithm 3**);

Get $\|\mathbf{d}_{i,j}\|^2$ for all pairs of sampled nodes and record the maximum value;

Get a σ^2 value via (2.25);

Algorithm 3 Finding a More Precise Noise Range

Run a simulation with σ^2 found with $noise_{up}$ in (2.24);

Get $F_{i,T+1}$ via (2.12) for $1 \leq i \leq T$ and record its minimum;

Using the absolute value of the minimum as the new $noise_{up}$, the new noise range is $[-noise_{up}, noise_{up}]$;

We can have similar definitions for the boundary between the Chaos Range and the Extending Range σ_{ce}^2 and the boundary between the Disturbing Range and the Averaging Range σ_{da}^2, which are used in later simulations. The boundary σ_{ce}^2 should be such that $B_{i,T+1}$ for

the smallest nonzero $\|\mathbf{d}_{i,T+1}\|^2$ is greater than $noise_{up}$. With the exponential property, we get

$$2\eta e^{-\frac{\|\mathbf{d}\|^2_{min,nonzero}}{2\sigma^2_{ce}}} = noise_{up}$$

where $\|\mathbf{d}\|^2_{min,nonzero}$ denotes the smallest nonzero $\|\mathbf{d}_{i,T+1}\|^2$, or equivalently,

$$\sigma^2_{ce} = -\frac{\|\mathbf{d}\|^2_{min,nonzero}}{2\ln\frac{noise_{up}}{2\eta}}. \tag{2.26}$$

The boundary σ^2_{da} should result in $B_{i,T+1}$ for the largest $\|\mathbf{d}_{i,T+1}\|^2$ to be close to 2η. That is, σ^2_{da} satisfies

$$2\eta e^{-\frac{\|\mathbf{d}\|^2_{max}}{2\sigma^2_{da}}} = 2\eta(1 - closeness)$$

or equivalently,

$$\sigma^2_{da} = -\frac{\|\mathbf{d}\|^2_{max}}{2\ln(1 - closeness)}. \tag{2.27}$$

where *closeness* should be chosen as a small value which indicates how much the value is expected to be close to 2η. For example, *closeness* has to be within $(0, 0.5)$ to imply the value is closer to 2η than 0. The choice is somewhat arbitrary as long as it indicates nearness to 2η.

2.5 Simulations

In this section, we provide simulation results confirming some properties of $B_{i,j}$ and $F_{i,j}$, and show the performance of the proposed algorithm on four real datasets.

2.5.1 Performance Measure for SKG

When talking about the performance of SKG, we follow [33] and use *generalization normalized mean squared error (GNMSE)* as the metric. GNMSE is defined as

$$\text{GNMSE} = \frac{\|\mathbf{y}_{true} - \mathbf{y}_{pred}\|^2}{\|\mathbf{y}_{true}\|^2} \tag{2.28}$$

where $\mathbf{y}_{pred}$ and $\mathbf{y}_{true}$ are vectors whose elements are the predicted and the true nodal values for all tested nodes, respectively. In addition, we will use normalized true values and predictions to calculate GNMSE.

2.5.2 Real Datasets

We use four real datasets, the Temperature-Jan dataset, the Cora-Con dataset, the Email-EU-Core dataset, and the Wikipedia-Math-Daily dataset.

The Temperature-Jan Dataset

The Temperature-Jan dataset is a part of the Temperature dataset which is used in Section 1.3. Recall that the Temperature dataset contains the average monthly temperature information of 83 weather stations in Switzerland. Note that the Temperature dataset does not contain any graph. It has altitude information of the stations. We used the altitude information to create two graphs. The first graph was created in the same way as authors of [24] created their ground truth graph based on the altitudes of stations. That is, an edge exists between a pair of nodes only when the altitude difference between the corresponding stations is less than 300 meters. The second graph is the same except that weights of connected nodes are not 1 but follow $e^{-\Delta/300}$ where Δ corresponds to the absolute value of the altitude

difference between a pair of connected nodes. The Temperature-Jan dataset contains the monthly average temperature information of all the 83 stations in January during 1961-1990, and the created graph of the 83 stations during the same period. Note that the first graph is unweighted whereas the second one is weighted. The temperature for the stations is viewed as nodal values.

The Cora-Con Dataset

The Cora-Con dataset is part of the Cora dataset. The Cora dataset [2] contains a citation network of 2708 scientific papers each of which is categorized as one of seven topics in the field of machine learning. We view the papers as nodes. Note that the citation network is an unweighted directed graph where edges can point from a citing paper to a cited paper. Then, the (column) adjacency vector of a node is actually an indicator vector of whether the paper cites a list of papers. We assign an integer from $\{1, 2, ..., 7\}$ representing paper classes as nodal values. The Cora dataset contains 486 papers with no citing. That is, these nodes have the adjacency vector of $\mathbf{0}$. But they carry different nodal values. To avoid these nodes confusing the SKG model, we create the Cora-Con dataset by excluding the 486 nodes, remaining 2222 nodes and the related network.

The Email-EU-Core Dataset

The Email-EU-Core dataset [44] contains email communication among 1005 members in a European research institution. Every member belongs to one of 42 departments. We view the members as nodes and assign an integer from $\{1, 2, ..., 42\}$ representing their membership as their nodal values. The communication network is unweighted and directed.

The Wikipedia-Math-Daily Dataset

The Wikipedia-Math-Daily dataset is part of the Wikipedia-Math dataset [45]. The Wikipedia-Math dataset contains a weighted link network among 1068 Wikipedia pages about Mathematics topics. The web pages are viewed as nodes and the network is directed. Weights on the links denote relevance. The dataset also contains daily visits for those pages between 2019 and 2021 March, 731 days in total. The daily visits on any day can be used as ground truth. The Wikipedia-Math-Daily contains the directed weighted network and daily visits on March 16th, 2019.

2.5.3 Exponential Approximation of $B_{i,j}$

The exponential approximation property is one of the core assumptions for other properties of $B_{i,j}$ and $F_{i,j}$. We aim to compare the exponential approximation with practical distributions of $B_{i,j}$. The Temperature-Jan dataset is used where 40% of total nodes (33 nodes) are randomly selected as sampled nodes. Referencing nodes are the sampled nodes. The parameter D is 200. The learning rate η is set to 0.1. The Gaussian variance σ^2 is set to 10.

Following the SKG algorithm and the definition of $B_{i,j}$ in (2.5), $B_{i,j}$ values for all pairs of sampled nodes are calculated. We select $B_{i,j}$ values with $\|\mathbf{d}_{i,j}\|^2 = 15$ as an example and get their distribution. There are 21 qualified pairs of nodes and the corresponding similarity measure take values within $(0.085, 0.105)$. The sample mean is 0.0976, and the sample variance is 5.80×10^{-5}. Recall that the exponential approximation in (2.10) states that $B_{i,j}$ can be approximated as $2\eta e^{-\frac{\|\mathbf{d}_{i,j}\|^2}{2\sigma^2}} = 0.094$ when $\|\mathbf{d}_{i,j}\|^2 = 15$. Comparing with the potential range $(0, 0.2]$ for $B_{i,j}$ with no $\|\mathbf{d}_{i,j}\|^2$ constraints, we could say the exponential approximation is valid.

2.5.4 Conformity Property and the Impact of σ^2

The conformity property is at the core of the analysis of the impact of σ^2. We show $B_{i,j}$ and $F_{i,j}$ behavior with σ^2 in different ranges using the Temperature-Jan dataset. Again, 40% of total nodes are randomly chosen as sampled nodes which are also referencing nodes. The parameters are $D = 200$, $\eta = 0.1$, and $E = 3$.

We first make sure the boundaries between adjacent ranges. Checking with the Temperature-Jan dataset, we know $\|\mathbf{d}\|^2_{min,nonzero} = 1$, and $\|\mathbf{d}\|^2_{max} = 27$. With $noise_{up}$ in (2.24) and $closeness = 0.1$, we get $\sigma^2_{ce} = 0.22$, $\sigma^2_{ed} = 5.87$, and $\sigma^2_{da} = 135$. Fig. 2.4 shows examples of $B_{i,T+1}$ and $F_{i,T+1}$ with σ^2 in different ranges. The conformity between $B_{i,j}$ and $F_{i,j}$ can be observed from the figures. Additionally, Fig. 2.5 displays an example of α_{T+1-i} values when σ^2 is in its Disturbing Range.

Given detailed look, it is seen from Fig. 2.4(a) that when σ^2 is in its Chaos Range, $B_{i,T+1}$ values are around 0, and so do $F_{i,T+1}$ values. Fig. 2.4(b) verifies that when σ^2 is in its Extending Range, positive and negative values of $B_{i,T+1}$ become unbalanced, and some $F_{i,T+1}$ values are greatly larger than 0. The $B_{i,T+1}$ values are greater than 0 in Fig. 2.4(c), and $F_{i,T+1}$ for the penultimate node are relatively large although its $B_{i,T+1}$ is close to $\min\{B_{i,T+1}\}_{i=1}^{T}$. In Fig. 2.4(d), $B_{i,T+1}$ values are all close to 2η, and $F_{i,T+1}$ is roughly an exponential function with respect to i.

Fig. 2.6 plots predictions for the tested nodes under different σ^2 as well as their true nodal values. Predictions with $\sigma^2 = 0.1$ have the greatest error whereas predictions with $\sigma^2 = 300$ are almost the same. In summary, as σ^2 grows, predictions tend to get closer to their ground truth, while σ^2 grows furthermore than needed, predictions are roughly the same for different tested nodes for the Temperature-Jan dataset.

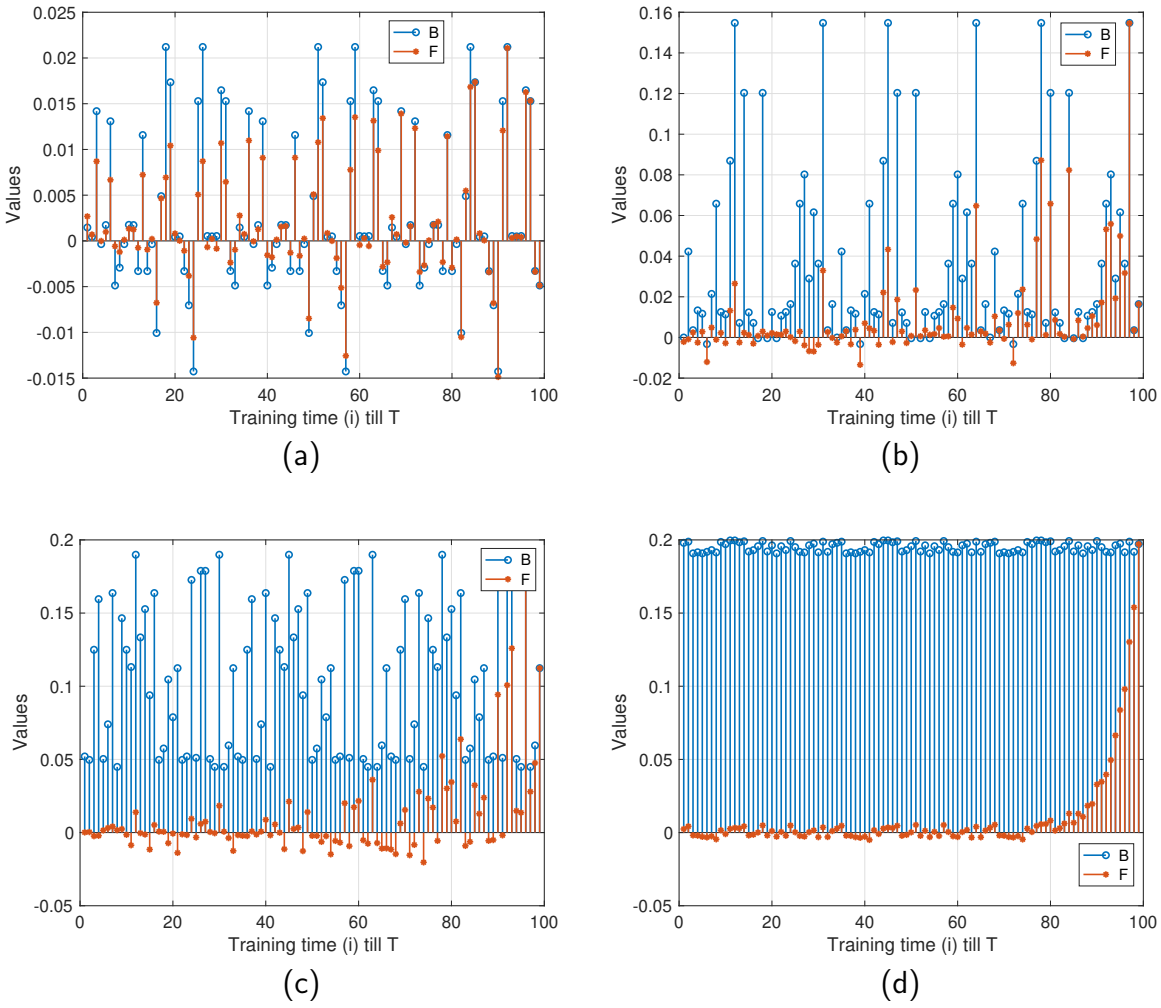

Figure 2.4: Values of $B_{i,T+1}$ and $F_{i,T+1}$ for a tested node in the Temperature-Jan dataset with different σ^2 values. (a) $\sigma^2 = 0.1$. (b) $\sigma^2 = 2$. (c) $\sigma^2 = 10$. (d) $\sigma^2 = 300$.

2.5.5 Performance of the Proposed Algorithm

To show the performance of the proposed algorithm, we compare the theoretical σ^2_{ed} value from Algorithm 2 with the best σ^2 found by simulations using the three real datasets. In each dataset, 40% of total nodes are randomly selected as the sampled nodes which are also referencing nodes. The $noise_{up}$ in (2.24) of $F_{i,j}$ is used. Simulation results will be denoted in blue solid curves and their corresponding proposed σ^2_{ed} will be denoted in red dotted line.

For unweighted graph in the Temperature-Jan dataset, the value of $\|\mathbf{d}\|^2_{max}$ is found to be 27,

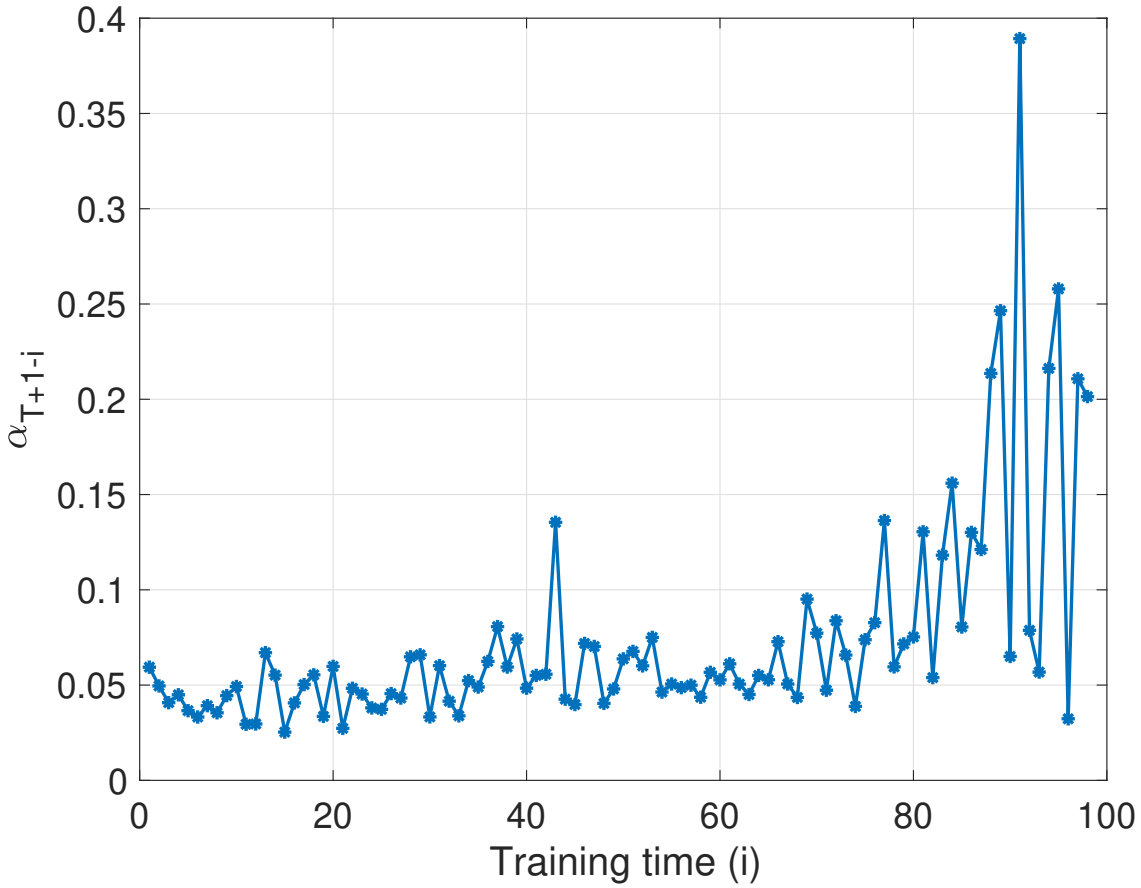

Figure 2.5: Values of α_{T+1-i} for a tested node in the Temperature-Jan dataset with $\sigma^2 = 10$.

and the theoretical result is $\sigma_{ed}^2 = 5.86$. For the weighted graph, the value of $\|\mathbf{d}\|_{max}^2$ is found to be 12.49 resulting in the theoretical result $\sigma_{ed}^2 = 2.08$. When finding the relationships between GNMSE and σ^2 by simulations, we set $E = 3$, $D = 200$, and $\eta = 0.1$. The results are shown in Fig. 2.7a and Fig. 2.7b, respectively. Note that the shown values of GNMSE are averaged over 50 repeated experiments.

For the Cora-Con dataset, the value of $\|\mathbf{d}\|_{max}^2$ is found out to be 8, resulting a theoretical value $\sigma_{ed}^2 = 1.05$. For simulations, we set $E = 3$, $D = M = 888$, and $\eta = 0.05$. The results are shown in Fig. 2.7c noting that the shown values of GNMSE are averaged over 30 repeated experiments. The GNMSE curve is not as smooth as in the previous case since fewer repeated experiments are carried out due to the larger network size and higher computational cost.

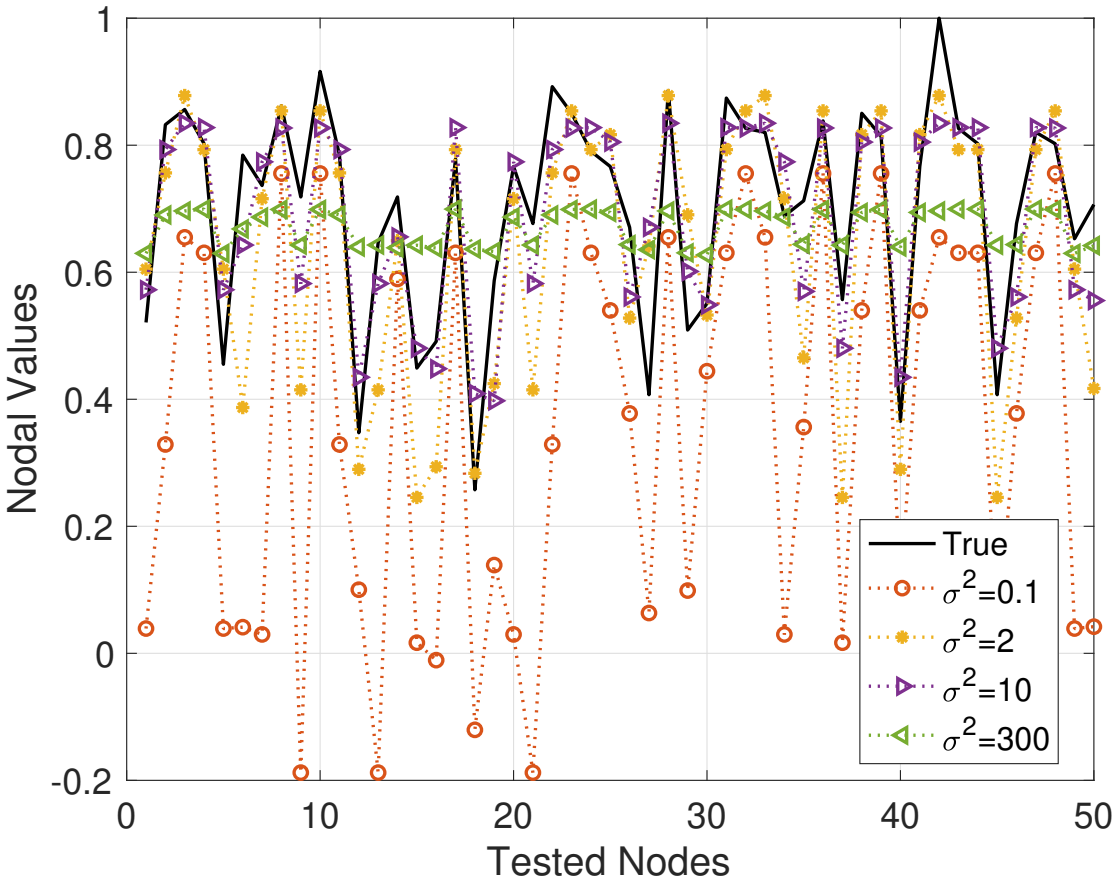

Figure 2.6: True nodal values and predicted values when $\sigma^2 = 0.1$, $\sigma^2 = 2$, $\sigma^2 = 10$, and $\sigma^2 = 300$ for tested nodes in the Temperature-Jan dataset.

For the Email-EU-Core dataset, the value of $\|\mathbf{d}\|^2_{max}$ is found out to be 107, resulting a theoretical value $\sigma^2_{ed} = 15.29$. For simulations, we set $E = 3$, $D = M = 403$, and $\eta = 0.05$. The results are shown in Fig. 2.7d. The shown values of GNMSE are averaged over 50 repeated experiments.

For the Wikipedia-Math-Daily dataset, the value of $\|\mathbf{d}\|^2_{max}$ is found out to be 671 followed by a theoretical value $\sigma^2_{ed} = 95.67$. For simulations, we set $E = 3$, $D = 500$, and $\eta = 0.03$. The comparison between the simulation result and the theoretical value is shown in Fig. 2.7e. The shown values of GNMSE are averaged over 50 repeated experiments.

2.6 Discussions

2.6.1 Complexity of the Proposed Algorithm

The majority of computational source for the proposed algorithm is used for calculating $\|\mathbf{d}_{i,T+1}\|^2$ for all pairs of sampled nodes. For a training set containing N sampled nodes, it would take $N(N-1)/2$ vector additions and inner products. Finding the largest $\|\mathbf{d}_{i,T+1}\|^2$ value can be done along with its calculation, and takes minor computational resource and memory resource.

2.6.2 Performance of the Proposed Algorithm

Impact From Dataset Statistics

It is noticed that the proposed algorithm finds the best σ^2 in terms of GNMSE for the Temperature-Jan dataset, whereas for the other three datasets, the best σ^2 is slightly larger than what is found via the proposed algorithm. That is partly because similar adjacency vectors leading to similar nodal values is not guaranteed in later datasets. We take the comparison between the Temperature-Jan dataset and the Cora-Con dataset as an example.

Let us first understand the relationship between similarity in nodal values and similarity in adjacency vectors in the two datasets. We use $\|\mathbf{d}_{i,j}\|^2$ to show dissimilarity of adjacency vectors and $|y_i - y_j|$ to show dissimilarity of nodal values for a pair of nodes. Notice the smaller the values, the more the similarity. Fig. 2.8 and Fig. 2.9 show $|y_i - y_j|$ versus $\|\mathbf{d}_{i,j}\|^2$ for the two datasets, respectively.

In Fig. 2.8, the horizontal axis takes discrete values because the graph is unweighted, and the vertical axis takes continuous values because the nodal values, i.e., temperature for the

stations, take continuous values. More importantly, it is seen that when $\|\mathbf{d}_{i,j}\|^2$ is small, the nodal value difference is also small. For instance, when $\|\mathbf{d}_{i,j}\|^2 = 5$ for a pair of nodes, $|y_i - y_j|$ may take a value within $[0, 0.35]$. Whereas, for a pair of nodes with $\|\mathbf{d}_{i,j}\|^2 = 15$, the nodal value difference is likely to be greater than 0.3 and could be up to 1. (The behavior for $\|\mathbf{d}_{i,j}\|^2 \geq 15$ corresponds to the fact that stations with higher altitudes have smaller temperature difference. Although such stations may largely vary in altitudes and thus adjacency vectors, they have relatively low temperature.) This property of the dataset is due to the fact that connected nodes have closer altitudes. Since connected nodes tend to have similar adjacency vectors in the graph, considering correlation between altitudes and temperature, it is expected that similar adjacency vectors tend to have closer nodal values (temperature).

In Fig. 2.9, the horizontal axis takes discrete values. The vertical axis also takes discrete values because the nodal labels in the dataset represent categories. It is seen that $|y_i - y_j|$ takes large values with nonnegligible probability even when $\|\mathbf{d}_{i,j}\|^2 = 0$. Recalling that the graph indicates citing behavior among papers, such a relationship between nodal values and adjacency vectors implies that the category of a citing paper is weakly correlated with the categories of the cited papers. That is, it is possible that two papers citing the same reference belong to different classes, and that two papers with different citing patterns are categorically the same. From the SKG model perspective, similarity in adjacency vectors may not lead to similarity in nodal values in the Cora-Con dataset.

Now we explain how such a relation between adjacency vectors and nodal values impacts the choice of best σ^2. Recall at the boundary between the Extending Range and the Disturbing Range, the efficient distance shown in Fig. 2.3 divides sampled nodes based on similarity in adjacency vectors, and the prediction can be considered as a weighted average of the inner nodes. For the Temperature-Jan dataset, the inner nodes have small nodal value difference from the tested node, and thus the prediction would be close. Whereas, for the Cora-Con

dataset, the predictions are precise for some tested nodes while they have large errors for the rest. However, these large errors can be reduced in the Disturbing Range. Recall at this range, predictions are weighted averages over all sampled nodes. Large errors at the boundary σ_{ed}^2 will become less in the Disturbing Range, but meanwhile, small errors may increase. Since GNMSE penalizes more on larger error, GNMSE performance could get better in the Disturbing Range than at σ_{ed}^2 for the Cora-Con dataset.

In summary, for datasets where similar adjacency vectors lead to dissimilar nodal values with nonnegligible probability, the resulting σ_{ed}^2 of the proposed algorithm is smaller than the best σ^2 which is in the Disturbing Range in terms of GNMSE.

Impact From Other Hyperparameters

We would like to note that our analysis applies for ideal cases where the number of random features D is sufficiently large and the learning rate η and the number of epochs E are properly chosen.

If D is not large enough, the validity of the exponential approximation of $B_{i,j}$ would weaken, leading to larger GNMSE values. The impact from D applies to all σ^2 such that the chosen D does not affect the best σ^2 theoretically. Our proposed algorithm provides σ^2 which is near-optimal, σ_{ed}^2 is affected by D without **Algorithm 3**. However, the impact of D decays as D increases due to the $\log(\cdot)$ function in the denominator of (2.25). In practice, D is usually more than tens or hundreds and thus the impact of D on σ_{ed}^2 is limited.

The parameters η and E should be jointly chosen such that $\sum_{i=1}^{EN} F_{i,EN+1} \approx 1$. Note that η cannot be very large to avoid parameter explosion. When η becomes smaller, E should be increased accordingly. In this case, our analysis applies and σ_{ed}^2 is close to optimum. If E is not sufficient either, σ_{ed}^2 would not be ideal because the weighting property of $F_{i,j}$ is invalid. In this case, the best σ^2 should be much larger than σ_{ed}^2 to mitigate the biased sum of the

contribution weights.

2.6.3 Similarity Transfer of SKG

We refer the reader to Figs. 2.7a–2.7d. Note that GNMSE values vary considerably among the three datasets. Thus, it is believed that SKG has better performance on some datasets than others. This can be understood via the weighting property together with the characteristic of a dataset. With σ_{ed}^2, predictions are weighted averages of a group of nodes. The weights are acquired from adjacency vectors and applied for nodal values. If similar adjacency vectors do not lead to similar nodal values, predictions would have large errors. If similar adjacency vectors lead to dissimilar nodal values with nonnegligible probability in a dataset, the SKG performance would be less ideal on the dataset. This is clear when comparing SKG performance between Temperature-Jan dataset on which GNMSE could be lower than 0.05 and the Cora-Con dataset on which GNMSE values are above 0.35. From the comparison, we concluded that SKG with a Gaussian kernel assumes that similarity of adjacency vectors leads to similarity of nodal values.

2.6.4 Extension of the Analysis

We used SKG with a Gaussian kernel to illustrate our analysis framework based on similarity measures and contribution weights. The analysis framework is not constrained in the cases of Gaussian kernels. In fact, it is applicable to all shift-invariant kernels [23]. Specifically speaking, as we mentioned before, the exponential approximation is the same expression as the random feature approximation but in a reverse order. So, any shift-invariant kernel that has a Fourier transform can easily build its similarity measure. In what follows, the proposed properties of contribution weights apply no matter what kernel is used because they are built only on the similarity measure. How to find a suitable kernel parameter based

on the framework is discussed in the following chapter.

Additionally, our analysis framework can be helpful for other algorithms which have a (shift-invariant-)kernel-based learning model. The framework is generalized to be applicable on general KRG-RFF models in the following chapter.

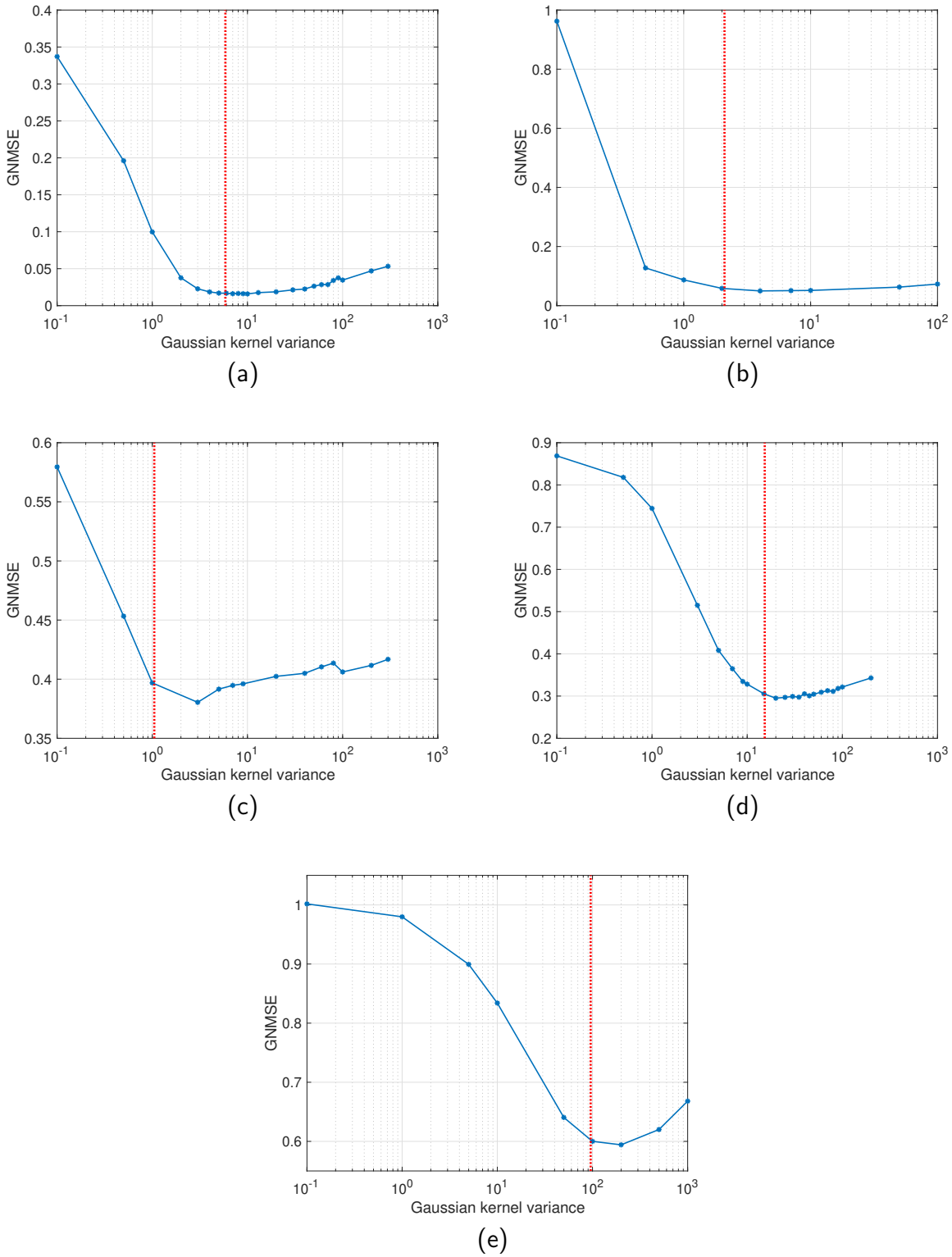

Figure 2.7: The GNMSE values with respect to the Gaussian kernel variance σ^2 for different datasets and graphs. (a) For the Temperature-Jan dataset with the unweighted graph. (b) For the Temperature-Jan dataset with the weighted graph. (c) For the Cora-Con dataset. (d) For the Email-EU-Core dataset. (e) For the Wiki-Math-Daily dataset.

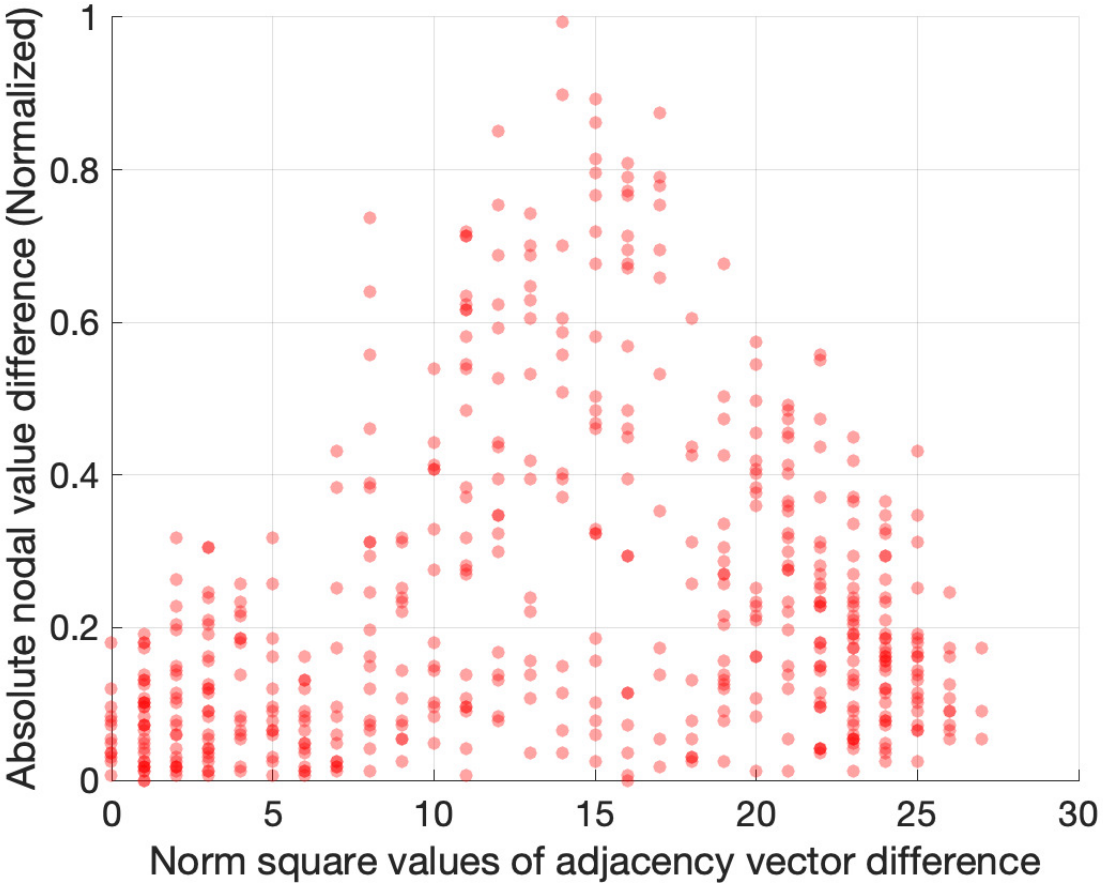

Figure 2.8: Scatter plot of the relationship between norm square of adjacency vector difference $\|\mathbf{d}_{i,j}\|^2$ and absolute difference between nodal values $|y_i - y_j|$ for all pairs of sampled nodes in the Temperature-Jan dataset.

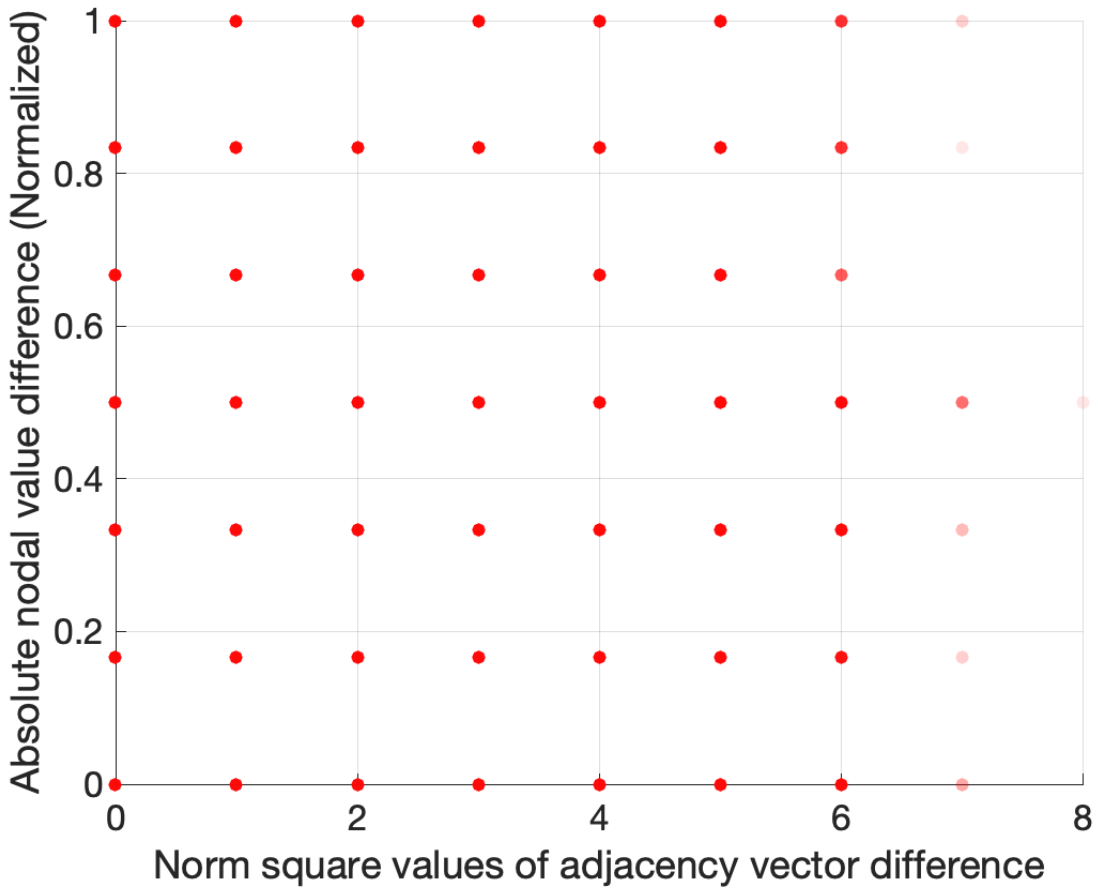

Figure 2.9: Scatter plot of the relationship between norm square of adjacency vector difference $\|\mathbf{d}_{i,j}\|^2$ and absolute difference between nodal values $|y_i - y_j|$ for all pairs of sampled nodes in the Cora-Con dataset.

Chapter 3

A General Framework

Kernel-based regression over graphs with random Fourier features (KRG-RFF) is an existing simple yet powerful algorithm that can perform graph signal prediction when the input signal is not limited to a graph signal. This chapter aims to deal with the configuration problem for KRG-RFF models to achieve (sub-)optimal performance given a dataset. Specifically, we extend the aforementioned parameter analysis framework and establish a generalized framework which expresses model predictions as weighted sums of training observations. With the new framework, we can get an understanding of the influences from the hyper parameters in the KRG-RFF model by studying how those parameters affect the observation weights. Based on this understanding of parameter influence, we propose corresponding methods to find suitable values for different parameters. Simulation results on real datasets are provided to confirm the validity of our understanding of the model parameters and to show the performance of the proposed parameters.

3.1 Introduction

Graph signals can change with time and other inputs, and thus, graph signal prediction [46, 47, 48, 30, 32] has been a topic in GSP. To solve the prediction problem, authors in [46, 47, 48] establish models using historical graph signal observations and the graph structure. That is, both the input and the output for the models are graph signals. On the contrary, models in [30, 32] can take more general inputs while enforcing the output to be a graph signal. More specifically, to model a nonlinear relationship between the input and the output for scenarios where the input is agnostic, i.e., the input is not limited to graph signals, authors of [30] incorporate kernels in a regression model on graphs and propose the kernel regression method over graphs (KRG). However, as the number of training input-output pairs and the network size increase, the computational burden becomes prohibitive. To deal with the scaling issue of the KRG algorithm, [32] proposes an iterative method, kernel regression over graphs with random Fourier features (KRG-RFF), by applying random Fourier feature (RFF) approximation [23] for shift-invariant kernels.

As seen in the Gradraker case, the KRG-RFF performance is usually affected by the hyper parameters of the model which should be chosen prior to the iterative training and will be fixed when the model is being used. Authors of [32] provide convergence ranges for learning rates in the first-order sense and the second order sense, and show in simulations the influence of the batch size and the number of random Fourier features (RFFs). The convergence analysis on a model using the graph kernel least mean squares based on random Fourier features (GKLMS-RFF which is similar to KRG-RFF) with respect to learning rates can be observed in [49]. The influence of the Gaussian kernel on graph-based adaptive learning using random feature approximation with multiple kernels (Gradraker) models [33] is presented in our previous work [50]. Those works include analysis for some parameters theoretically or via simulation. But they do not focus on the kernel-based learning model with RFF approximation and perform parameter analysis systematically. We aim to establish

a framework to understand influences from all parameters in those learning models. To do so, we will extend the work in Chapter 2 [50]. The work establishes an analyzing framework based on two intermediate variables. In the framework, the influence of the Gaussian kernel variance on predictions can be understood via investigating the influence of the Gaussian variance on the two intermediate variables. In this chapter, we will generalize the two intermediate variables. In addition, we will introduce a third variable to specifically deal with the batch training in KRG-RFF. With understandings of parameter influences on model performance, we are able to find suitable values for different hyper parameters.

The KRG-RFF model has been reviewed in Section 1.2. Comparing the mathematical expressions for general KRG-RFF models and those for Gradraker models, we could find that although their inputs and outputs have different physical meanings, their mathematical forms are intrinsically the same. Thus, the generalized framework with three intermediate variables is illustrated in Section 3.2. Understandings of influences from different hyper parameters are presented in Section 3.3 based on the proposed framework. Simulation results on real datasets are provided in Section 3.4.

3.2 Mathematical Tools

To achieve a better performance by choosing suitable parameters configuring the KRG-RFF model, parameter analysis is inevitable. However, due to the recursion nature of the training process, it is hard to rewrite model prediction errors with respect to model parameters directly. So, we manage to rewrite model predictions as weighted sum of training observations and study the influence of parameters on weights of observations.

The two variables in the previous chapter, the *contribution weights* and the *similarity measure*, are generalized in this chapter to analyze KRG-RFF models. Recall that the contri-

bution weights are the actual weights of training observations for predictions in the training phase, and the similarity measure is an intermediate variable needed for the definition of contribution weights. In addition, a third variable, the *tailored RF*, is introduced to deal with the batch-based optimization in KRF-RFF models specifically.

We would like to note that, although the KRG-RFF model can be updated when the model is in use as long as the ground truth is available, for simplicity in analysis, we assume that there exists a time instance N such that inputs that come before and at N are with ground truth and inputs that come after N are without ground truth. That is, N is the training duration and also the threshold for the training and the predicting phases. Inputs used in the predicting phase are all viewed as inputs at $N+1$ without updating the model. The analysis for this basic case can be easily extended to more general cases.

In addition, for simplicity in the analysis, we set the factor β of the smoothness term to be small enough that the smoothness term can be ignored. This is a valid assumption for noiseless cases. Because, in noiseless cases, according to the objective function in (1.8), observations are ground truth and thus smoothness term introduces bias when β is large. For the following analysis, β is set to be 0.

3.2.1 Similarity Measure

For training time instances m and n such that $N_b \leq n < m \leq N$, we define the similarity measure

$$\mathbf{B}_{m,n} = \frac{\mu}{N_b} \mathbf{Z}_m \mathbf{Z}_n^\top. \tag{3.1}$$

Substituting (1.11) in (3.1), we get

$$\mathbf{B}_{m,n} = \frac{\mu}{N_b} \begin{bmatrix} z^\top (\mathbf{x}_{m-N_b+1}) \\ \vdots \\ z^\top (\mathbf{x}_m) \end{bmatrix} [z (\mathbf{x}_{n-N_b+1}), \cdots, z (\mathbf{x}_n)]$$

as an N_b-by-N_b matrix whose (i, j)-th element is

$$[\mathbf{B}_{m,n}]_{i,j} = \frac{\mu}{N_b} z^\top (\mathbf{x}_{m-N_b+i}) z (\mathbf{x}_{n-N_b+j}) \approx \frac{\mu}{N_b} \kappa (\mathbf{x}_{m-N_b+i}, \mathbf{x}_{n-N_b+j})$$

where the approximation holds when the number of RFs D is sufficiently large. Different elements in a $\mathbf{B}$ matrix consider inputs at different time instances in a batch, but all elements are approximations of kernel values with the same factor $\frac{\mu}{N_b}$. Recalling that kernels are used for measuring similarity between inputs, $\mathbf{B}$ matrices are inherently used as a similarity measure.

When the batch size N_b is 1, the $\mathbf{B}$ variable is reduced to a scalar

$$B_{m,n} = \frac{\mu}{N_b} z^\top (\mathbf{x}_m) z (\mathbf{x}_n)$$

which is the same as the definition of the similarity measure in Chapter 2.

3.2.2 Contribution Weight

The contribution weight has definition only at time instances being multiples of the refreshing period δ.

Let γ and l be positive integers such that $N_b \leq l\delta < \gamma\delta \leq N$, we define the contribution

weight $\mathbf{F}$ as

$$\mathbf{F}_{\gamma\delta,l\delta} = \mathbf{B}_{\gamma\delta,l\delta} \tag{3.2}$$

for $\gamma = l + 1$, and

$$\mathbf{F}_{\gamma\delta,l\delta} = (1-\mu\alpha)^{\gamma-l-1}\mathbf{B}_{\gamma\delta,l\delta} - \sum_{k=l+1}^{\gamma-1}(1-\mu\alpha)^{\gamma-k-1}\mathbf{B}_{\gamma\delta,k\delta}\mathbf{F}_{k\delta,l\delta} \tag{3.3}$$

for $\gamma > l + 1$.

Like $\mathbf{B}$ matrices, $\mathbf{F}$ matrices are also of size $N_b \times N_b$. The definition of $\mathbf{F}$ matrices retains the recursion nature of the training process. We will use the following properties to gain a concrete idea of $\mathbf{F}$ matrices.

Weighting Property

Recalling that the smoothness term in (1.8) is ignored, the following claim shows $\mathbf{F}$ matrices are used as weights of batch observations for prediction during the training phase.

Claim 3.1. *Let γ be a positive integer such that $N_b \leq \gamma\delta \leq N$. Given definitions in (3.1), (3.2), and (3.3), we can rewrite predictions during training phase $\mathbf{Y}_{\gamma\delta}$ in terms of historical batch observations $\mathbf{T}_{l\delta}, l = \gamma_0, \cdots, \gamma - 1$ as*

$$\mathbf{Y}_{\gamma\delta} = \sum_{l=\gamma_0}^{\gamma-1}\mathbf{F}_{\gamma\delta,l\delta}\mathbf{T}_{l\delta} \tag{3.4}$$

where γ_0 is the least integer such that $\gamma_0\delta \geq N_b$.

Proof. Please refer to the proof of **Claim 3.2** first.

Letting γ_0 be the least integer such that $\gamma_0\delta \geq N_b$, the prediction at the training time

instance $\gamma\delta$ is

$$\mathbf{Y}_{\gamma\delta} = \mathbf{Z}_{\gamma\delta}\mathbf{H}_{\gamma\delta} = \mathbf{Z}_{\gamma\delta} \sum_{l=\gamma_0}^{(\gamma-1)\delta} \mathbf{G}_{\gamma\delta,l\delta}\mathbf{T}_{l\delta} = \sum_{l=\gamma_0}^{(\gamma-1)\delta} \mathbf{Z}_{\gamma\delta}\mathbf{G}_{\gamma\delta,l\delta}\mathbf{T}_{l\delta}$$

where the second equality follows **Claim 3.2**. Recalling the featuring property of $\mathbf{G}$, i.e, $\mathbf{Z}_{\gamma\delta}\mathbf{G}_{\gamma\delta,l\delta} = \mathbf{F}_{\gamma\delta,l\delta}$, we have $\mathbf{Y}_{\gamma\delta} = \sum_{l=\gamma_0}^{(\gamma-1)\delta} \mathbf{F}_{\gamma\delta,l\delta}\mathbf{T}_{l\delta}$. ■

Claim 3.1 shows batch prediction $\mathbf{Y}_{\gamma\delta}$ during the training phase is actually the weighted sum of previous batch observations $\mathbf{T}_{l\delta}, l = \gamma_0, \cdots, \gamma - 1$ where $\mathbf{F}$ matrices are the weights. This is called the weighting property of $\mathbf{F}$ matrices.

Notice that (3.4) applies on the training phase only. Because, during the training phase, the model takes multiple inputs at a time; whereas during the predicting phase, we focus on how the model works given an input. Still, (3.4) gives insights on prediction in the predicting phase since predictions during the training and predicting phases are essentially the same. More specifically, the expression of predictions $\mathbf{y}_{N+1}$ during the predicting phase can be generalized from $\mathbf{y}_{\gamma\delta}$ which is the last row of $\mathbf{Y}_{\gamma\delta}$ acquired via (3.4).

Conformity Property with B Matrices

Due to the recursion definition of $\mathbf{F}$ matrices, it is hard to get closed-form expressions for $\mathbf{F}$. Fortunately, we find $\mathbf{B}$ matrices and $\mathbf{F}$ matrices with the same indices behave similarly to some extent. We illustrate the phenomenon through induction in the following.

For $l = \gamma - 1$, we have

$$\mathbf{F}_{\gamma\delta,l\delta} = \mathbf{F}_{\gamma\delta,(\gamma-1)\delta} = \mathbf{B}_{\gamma\delta,(\gamma-1)\delta}.$$

For $l = \gamma - 2$, with $\mathbf{F}_{(\gamma-1)\delta,(\gamma-2)\delta} = \mathbf{B}_{(\gamma-1)\delta,(\gamma-2)\delta}$, we have

$$\mathbf{F}_{\gamma\delta,l\delta} = \mathbf{F}_{\gamma\delta,(\gamma-2)\delta} = (1-\mu\alpha)\mathbf{B}_{\gamma\delta,(\gamma-2)\delta} - \mathbf{B}_{\gamma\delta,(\gamma-1)\delta}\mathbf{B}_{(\gamma-1)\delta,(\gamma-2)\delta}$$

whose (i,j)-th element is

$$\begin{aligned}\left[\mathbf{F}_{\gamma\delta,(\gamma-2)\delta}\right]_{i,j} =& (1-\mu\alpha)\left[\mathbf{B}_{\gamma\delta,(\gamma-2)\delta}\right]_{i,j} - \sum_{k=1}^{N_b}\left[\mathbf{B}_{\gamma\delta,(\gamma-1)\delta}\right]_{i,k}\left[\mathbf{B}_{(\gamma-1)\delta,(\gamma-2)\delta}\right]_{k,j} \\ =& (1-\mu\alpha)\left[\mathbf{B}_{\gamma\delta,(\gamma-2)\delta}\right]_{i,j}\cdot\left[1 - \sum_{k=1}^{N_b}\frac{\left[\mathbf{B}_{\gamma\delta,(\gamma-1)\delta}\right]_{i,k}\left[\mathbf{B}_{(\gamma-1)\delta,(\gamma-2)\delta}\right]_{k,j}}{(1-\mu\alpha)\left[\mathbf{B}_{\gamma\delta,(\gamma-2)\delta}\right]_{i,j}}\right].\end{aligned} \tag{3.5}$$

When the number of RF D is sufficiently large, we consider the RFF approximation in (1.5) and get the following for the fractions in the summation

$$\begin{aligned}\frac{\left[\mathbf{B}_{\gamma\delta,(\gamma-1)\delta}\right]_{i,k}\left[\mathbf{B}_{(\gamma-1)\delta,(\gamma-2)\delta}\right]_{k,j}}{(1-\mu\alpha)\left[\mathbf{B}_{\gamma\delta,(\gamma-2)\delta}\right]_{i,j}} \approx& \frac{\frac{\mu}{N_b}\xi_{\gamma,i,\gamma-1,k}\frac{\mu}{N_b}\xi_{\gamma-1,k,\gamma-2,j}}{(1-\mu\alpha)\frac{\mu}{N_b}\xi_{\gamma,i,\gamma-2,j}} \\ =& \frac{1}{N_b}\frac{\mu}{1-\mu\alpha}\frac{\xi_{\gamma,i,\gamma-1,k}\xi_{\gamma-1,k,\gamma-2,j}}{\xi_{\gamma,i,\gamma-2,j}}\end{aligned} \tag{3.6}$$

with small approximation errors. Note that we use

$$\xi_{\gamma,i,\gamma-1,k} = \kappa\left(\mathbf{x}_{\gamma\delta-N_b+i}, \mathbf{x}_{(\gamma-1)\delta-N_b+k}\right)$$

for notational simplicity. Substituting the fraction approximation in (3.6) to (3.5), we get

$$\left[\mathbf{F}_{\gamma\delta,(\gamma-2)\delta}\right]_{i,j} \approx (1-\mu\alpha)\left(1 - a_{2,\gamma,i,j}\right)\left[\mathbf{B}_{\gamma\delta,(\gamma-2)\delta}\right]_{i,j} \tag{3.7}$$

where

$$a_{2,\gamma,i,j} = \frac{\mu}{1-\mu\alpha}\cdot\frac{1}{N_b}\sum_{k=1}^{N_b}\frac{\xi_{\gamma,i,\gamma-1,k}\xi_{\gamma-1,k,\gamma-2,j}}{\xi_{\gamma,i,\gamma-2,j}}.$$

Notice that the sample-averaged fraction a is with respect to the batch index γ, the element position (i, j), and the time difference which is 2 in this case.

The approximation in (3.7) implies elements in $\mathbf{F}_{\gamma\delta,(\gamma-2)\delta}$ can be large or small according to elements in $\mathbf{B}_{\gamma\delta,(\gamma-2)\delta}$ in the same position. That is called the conformity property between $\mathbf{F}_{\gamma\delta,(\gamma-2)\delta}$ and $\mathbf{B}_{\gamma\delta,(\gamma-2)\delta}$.

For $l = \gamma - 3$, with $\mathbf{F}_{(\gamma-2)\delta,(\gamma-3)\delta} = \mathbf{B}_{(\gamma-2)\delta,(\gamma-3)\delta}$ and (3.7), we get (i, j)-th element in $\mathbf{F}_{\gamma\delta,(\gamma-3)\delta}$ as

$$\begin{aligned}\left[\mathbf{F}_{\gamma\delta,(\gamma-3)\delta}\right]_{i,j} \approx & (1-\mu\alpha)^2 \left[\mathbf{B}_{\gamma\delta,(\gamma-3)\delta}\right]_{i,j} \\ & - (1-\mu\alpha) \sum_{k=1}^{N_b} \left[\mathbf{B}_{\gamma\delta,(\gamma-2)\delta}\right]_{i,k} \left[\mathbf{B}_{(\gamma-2)\delta,(\gamma-3)\delta}\right]_{k,j} \\ & - (1-\mu\alpha) \sum_{k=1}^{N_b} (1 - a_{2,\gamma-1,k,j})\, p_k\end{aligned}$$

where $p_k = \left[\mathbf{B}_{\gamma\delta,(\gamma-1)\delta}\right]_{i,k} \left[\mathbf{B}_{(\gamma-1)\delta,(\gamma-3)\delta}\right]_{k,j}$ Taking $(1-\mu\alpha)^2 \left[\mathbf{B}_{\gamma\delta,(\gamma-3)\delta}\right]_{i,j}$ out of each term, we have

$$\begin{aligned}\left[\mathbf{F}_{\gamma\delta,(\gamma-3)\delta}\right]_{i,j} \approx & (1-\mu\alpha)^2 \left[\mathbf{B}_{\gamma\delta,(\gamma-3)\delta}\right]_{i,j} \\ & \cdot \frac{1}{N_b} \sum_{k=1}^{N_b} \left[1 - a_{3,\gamma,i,j,k,\gamma-2} - (1 - a_{2,\gamma-1,k,j})\, a_{3,\gamma,i,j,k,\gamma-1}\right]\end{aligned}$$

where

$$\begin{aligned}a_{3,\gamma,i,j,k,\gamma-2} = & \frac{\mu}{1-\mu\alpha} \frac{\xi_{\gamma,i,\gamma-2,k}\xi_{\gamma-2,k,\gamma-3,j}}{\xi_{\gamma,i,\gamma-3,j}} \\ a_{3,\gamma,i,j,k,\gamma-1} = & \frac{\mu}{1-\mu\alpha} \frac{\xi_{\gamma,i,\gamma-1,k}\xi_{\gamma-1,k,\gamma-3,j}}{\xi_{\gamma,i,\gamma-3,j}}.\end{aligned}$$

Notice that $a_{3,\gamma,i,j,k,\gamma-2}$, $a_{2,\gamma-1,k,j}$, and $a_{3,\gamma,i,j,k,\gamma-1}$ are mathematically the same. So, they are

in the same order and we can use

$$1 - a_{3,\gamma,i,j,k,\gamma-2} - (1 - a_{2,\gamma-1,k,j})\, a_{3,\gamma,i,j,k,\gamma-1} = (1 - a_{3,\gamma,i,j,k})^2$$

where $a_{3,\gamma,i,j,k}$ is in the same order as $a_{3,\gamma,i,j,k,\gamma-2}$, $a_{2,\gamma-1,k,j}$, and $a_{3,\gamma,i,j,k,\gamma-1}$. Then, we can rewrite $\left[\mathbf{F}_{\gamma\delta,(\gamma-3)\delta}\right]_{i,j}$ as

$$\left[\mathbf{F}_{\gamma\delta,(\gamma-3)\delta}\right]_{i,j} \approx (1 - \mu\alpha)^2 (1 - a_{3,\gamma,i,j})^2 \left[\mathbf{B}_{\gamma\delta,(\gamma-3)\delta}\right]_{i,j} \tag{3.8}$$

via the notation

$$(1 - a_{3,\gamma,i,j})^2 = \frac{1}{N_b} \sum_{k=1}^{N_b} [1 - a_{3,\gamma,i,j,k}]^2$$

where $a_{3,\gamma,i,j}$ is in the same order as $a_{3,\gamma,i,j,k}, k = 1, \cdots, N_b$.

The approximation in (3.8) is called the conformity property between $\mathbf{F}_{\gamma\delta,(\gamma-3)\delta}$ and $\mathbf{B}_{\gamma\delta,(\gamma-3)\delta}$. Similarly, it is not hard to conclude

$$[\mathbf{F}_{\gamma\delta,l\delta}]_{i,j} \approx [((1 - \mu\alpha)(1 - a_{\gamma-l,\gamma,i,j})]^{\gamma-l-1} [\mathbf{B}_{\gamma\delta,l\delta}]_{i,j}$$

which is the mathematical form of conformity property between corresponding $\mathbf{F}$ and $\mathbf{B}$ matrices.

The conformity property delivers a better idea of what $\mathbf{F}$ matrices are. Specifically, when the batch indices difference $\gamma - l$ is small, elements in $\mathbf{F}$ will be large if the corresponding elements in the respective $\mathbf{B}$ are large, and hence the corresponding training observations would be heavily weighted; otherwise they are lightly weighted. When the difference $\gamma - l$ is large, the corresponding graph signals are weighted lightly no matter what. In essence, $\mathbf{F}$ matrices display that the model transfers similarity in inputs to similarity in predictions.

When the batch size N_b is 1 and the parameter α is 0, the $\mathbf{F}$ variable is reduced to a scalar which is exactly the contribution weight introduced in [50].

3.2.3 Tailored RF

Recall that (3.4) in **Claim 3.1** applies on the training phase only because we focus on how the model works given an input during the testing phase whereas (3.4) outputs N_b predictions at a time. Although we could generalize the last row of $\mathbf{Y}$ to get prediction expressions for the predicting phase, we find a more convincing way to generate required prediction expressions. That is, we rewrite the trainable parameter $\mathbf{H}$ of the model. To do so, we introduce a new variable, the tailored RF, $\mathbf{G}$.

Like $\mathbf{F}$ matrices, $\mathbf{G}$ has definition only at time instances being multiples of the refreshing period δ. Let γ and l be positive integers such that $N_b \leq l\delta < \gamma\delta \leq N$, we define the tailored RF $\mathbf{G}$ as

$$\mathbf{G}_{\gamma\delta,l\delta} = \frac{\mu}{N_b}\mathbf{Z}^{\top}_{(\gamma-1)\delta} \tag{3.9}$$

for $l = \gamma - 1$ and

$$\mathbf{G}_{\gamma\delta,l\delta} = (1-\mu\alpha)^{\gamma-l-1}\frac{\mu}{N_b}\mathbf{Z}^{\top}_{l\delta} - \sum_{k=l+1}^{\gamma-1}(1-\mu\alpha)^{\gamma-k-1}\frac{\mu}{N_b}\mathbf{Z}^{\top}_{k\delta}\mathbf{F}_{k\delta,l\delta} \tag{3.10}$$

for $l < \gamma - 1$.

Notice that $\mathbf{G}$ matrices are of size $D \times N_b$. We can see from (3.9) clearly that $\mathbf{G}$ matrices are essentially some form of RF matrices $\mathbf{Z}$.

Weighting Property

The following claim shows $\mathbf{G}$ matrices are used as weights of observations in the trainable parameter $\mathbf{H}$ of the model.

Claim 3.2. *Let γ be a positive integer and $N_b \leq \gamma\delta \leq N$. Given definitions in (3.1), (3.2), (3.3), (3.9), and (3.10), we can rewrite the model parameter $\mathbf{H}_{\gamma\delta}$ in terms of historical observations $\mathbf{T}_{l\gamma}, l = \gamma_0, \cdots, \gamma - 1$ as*

$$\mathbf{H}_{\gamma\delta} = \sum_{l=\gamma_0}^{\gamma-1} \mathbf{G}_{\gamma\delta,l\delta}\mathbf{T}_{l\delta} \tag{3.11}$$

where γ_0 is the least integer such that $\gamma_0\delta \geq N_b$.

Proof. We use Mathematical Induction to prove the claim. Denoting γ_0 the least integer such that $\gamma_0\delta \geq N_b$, we initialize the trainable parameter $\mathbf{H}_{\gamma_0\delta} = \mathbf{0}_{D\times K}$. Then, the prediction at $n = \gamma_0\delta$ is $\mathbf{Y}_{\gamma_0\delta} = \mathbf{Z}_{\gamma_0\delta}\mathbf{H}_{\gamma_0\delta} = \mathbf{0}_{K\times 1}$ followed by $\mathbf{E}_{\gamma_0\delta} = \mathbf{T}_{\gamma_0\delta} - \mathbf{Y}_{\gamma_0\delta} = \mathbf{T}_{\gamma_0\delta}$. As a consequence, in the following time instance, $\mathbf{H}$ is updated and we get

$$\mathbf{H}_{\gamma_0\delta+1} = (1-\mu\alpha)\mathbf{H}_{\gamma_0\delta+1} + \frac{\mu}{N_b}\mathbf{Z}_{\gamma_0\delta}^{\top}\left(\mathbf{E}_{\gamma_0\delta}\right) = \frac{\mu}{N_b}\mathbf{Z}_{\gamma_0\delta}^{\top}\mathbf{T}_{\gamma_0\delta}$$

with the small β term to be ignored.

Notice that the model parameter will not change until $(\gamma_0+1)\delta$. For $n = (\gamma_0+1)\delta$, we have

$$\mathbf{H}_{(\gamma_0+1)\delta} = \frac{\mu}{N_b}\mathbf{Z}_{\gamma_0\delta}^{\top}\mathbf{T}_{\gamma_0\delta}$$

from which we can get the prediction

$$\mathbf{Y}_{(\gamma_0+1)\delta} = \mathbf{Z}_{(\gamma_0+1)\delta}\mathbf{H}_{(\gamma_0+1)\delta} = \frac{\mu}{N_b}\mathbf{Z}_{(\gamma_0+1)\delta}\mathbf{Z}_{\gamma_0\delta}^{\top}\mathbf{T}_{\gamma_0\delta} = \mathbf{B}_{(\gamma_0+1)\delta,\gamma_0\delta}\mathbf{T}_{\gamma_0\delta} = \mathbf{F}_{(\gamma_0+1)\delta,\gamma_0\delta}\mathbf{T}_{\gamma_0\delta}$$

considering definitions in (3.1) for $\mathbf{B}$, and (3.2) and (3.3) for $\mathbf{F}$. Then,

$$\begin{aligned}\mathbf{H}_{(\gamma_0+1)\delta+1} =&(1-\mu\alpha)\frac{\mu}{N_b}\mathbf{Z}_{\gamma_0\delta}^\top\mathbf{T}_{\gamma_0\delta}+\frac{\mu}{N_b}\mathbf{Z}_{(\gamma_0+1)\delta}^\top\left(\mathbf{T}_{(\gamma_0+1)\delta}-\mathbf{F}_{(\gamma_0+1)\delta,\gamma_0\delta}\mathbf{T}_{\gamma_0\delta}\right)\\ =&\frac{\mu}{N_b}\mathbf{Z}_{(\gamma_0+1)\delta}^\top\mathbf{T}_{(\gamma_0+1)\delta}+\left[(1-\mu\alpha)\frac{\mu}{N_b}\mathbf{Z}_{\gamma_0\delta}^\top-\frac{\mu}{N_b}\mathbf{Z}_{(\gamma_0+1)\delta}^\top\mathbf{F}_{(\gamma_0+1)\delta,\gamma_0\delta}\right]\mathbf{T}_{\gamma_0\delta}.\end{aligned}$$

The model parameter $\mathbf{H}$ will remain still till $n=(\gamma_0+2)\delta$. At this time instance, we have

$$\mathbf{H}_{(\gamma_0+2)\delta+1}=\frac{\mu}{N_b}\mathbf{Z}_{(\gamma_0+1)\delta}^\top\mathbf{T}_{(\gamma_0+1)\delta}+\left[(1-\mu\alpha)\frac{\mu}{N_b}\mathbf{Z}_{\gamma_0\delta}^\top-\frac{\mu}{N_b}\mathbf{Z}_{(\gamma_0+1)\delta}^\top\mathbf{F}_{(\gamma_0+1)\delta,\gamma_0\delta}\right]\mathbf{T}_{\gamma_0\delta}.$$

By checking the expressions for $\mathbf{H}_{\gamma_0\delta}$, $\mathbf{H}_{(\gamma_0+1)\delta}$, and $\mathbf{H}_{(\gamma_0+2)\delta}$, we find the claimed statement holds true for the three. Now, assume $\exists\gamma_1$ such that for $\gamma_0\le\gamma\le\gamma_1$, $\mathbf{H}_{\gamma\delta}=\sum_{l=\gamma_0}^{\gamma-1}\mathbf{G}_{\gamma\delta,l\delta}\mathbf{T}_{l\delta}$ with $\mathbf{G}_{\gamma\delta,l\delta}$ defined in (3.9) and (3.10), $\mathbf{B}_{\gamma\delta,l\delta}$ defined in (3.1), and $\mathbf{F}_{\gamma\delta,l\delta}$ defined in (3.2) and (3.3), respectively, for $\gamma_0\le l<\gamma\le\gamma_1$. Then, we would get

$$\mathbf{H}_{\gamma_1\delta+1}=(1-\mu\alpha)\sum_{l=\gamma_0}^{\gamma_1-1}\mathbf{G}_{\gamma_1\delta,l\delta}\mathbf{T}_{l\delta}+\frac{\mu}{N_b}\mathbf{Z}_{\gamma_1\delta}^\top\left(\mathbf{T}_{\gamma_1\delta}-\mathbf{Z}_{\gamma_1\delta}\sum_{l=\gamma_0}^{\gamma_1-1}\mathbf{G}_{\gamma_1\delta,l\delta}\mathbf{T}_{l\delta}\right).$$

Noticing that $\mathbf{Z}_{\gamma_1\delta}\mathbf{G}_{\gamma_1\delta,l\delta}=\mathbf{F}_{\gamma_1\delta,l\delta}$, we can rewrite $\mathbf{H}_{\gamma_1\delta+1}$ as

$$\mathbf{H}_{\gamma_1\delta+1}=\frac{\mu}{N_b}\mathbf{Z}_{\gamma_1\delta}^\top\mathbf{T}_{\gamma_1\delta}+\sum_{l=\gamma_0}^{\gamma_1-1}\left[(1-\mu\alpha)\mathbf{G}_{\gamma_1\delta,l\delta}-\frac{\mu}{N_b}\mathbf{Z}_{\gamma_1\delta}^\top\mathbf{F}_{\gamma_1\delta,l\delta}\right]\mathbf{T}_{l\delta}.$$

Since $\mathbf{H}$ will remain unchanged till $n=(\gamma_1+1)\delta$, we have

$$\mathbf{H}_{(\gamma_1+1)\delta}=\sum_{l=\gamma_0}^{\gamma_1}\mathbf{G}_{(\gamma_1+1)\delta,l\delta}\mathbf{T}_{l\delta}$$

by denoting $\mathbf{G}_{(\gamma_1+1)\delta,\gamma_1\delta}=\frac{\mu}{N_b}\mathbf{Z}_{\gamma_1\delta}^\top$ and

$$\mathbf{G}_{(\gamma_1+1)\delta,l\delta}=(1-\mu\alpha)\mathbf{G}_{\gamma_1\delta,l\delta}-\frac{\mu}{N_b}\mathbf{Z}_{\gamma_1\delta}^\top\mathbf{F}_{\gamma_1\delta,l\delta}$$

$$
\begin{aligned}
=&(1-\mu\alpha)^{\gamma_1+1-l-1}\frac{\mu}{N_b}\mathbf{Z}_{l\delta}^{\top}-\sum_{k=l+1}^{\gamma_1-1}(1-\mu\alpha)^{\gamma_1+1-k-1}\frac{\mu}{N_b}\mathbf{Z}_{k\delta}^{\top}\mathbf{F}_{k\delta,l\delta}-\frac{\mu}{N_b}\mathbf{Z}_{\gamma_1\delta}^{\top}\mathbf{F}_{\gamma_1\delta,l\delta}\\
=&(1-\mu\alpha)^{\gamma_1+1-l-1}\frac{\mu}{N_b}\mathbf{Z}_{l\delta}^{\top}-\sum_{k=l+1}^{\gamma_1}(1-\mu\alpha)^{\gamma_1+1-k-1}\frac{\mu}{N_b}\mathbf{Z}_{k\delta}^{\top}\mathbf{F}_{k\delta,l\delta}
\end{aligned}
$$

which proves the claimed statement holds true for γ_1+1 using definitions of $\mathbf{B}_{\gamma\delta,l\delta}$ in (3.1) and $\mathbf{F}_{\gamma\delta,l\delta}$ in (3.2) and (3.3) for $\gamma_0 \leq l < \gamma \leq \gamma_1+1$. ■

According to **Claim 3.2**, we see that the model $\mathbf{H}$ can be seen as weighted sum of previous observations where $\mathbf{G}$ matrices play weights. That is called the weighting property of $\mathbf{G}$. Although (3.11) is for the training phase, it can be easily extended to the predicting phase. Denoting γ_N the largest integer such that $\gamma_N\delta \leq N$, we have the well-trained model

$$
\mathbf{H}_{N+1}=\sum_{l=\gamma_0}^{\gamma_N}\mathbf{G}_{(\gamma_N+1)\delta,l\delta}\mathbf{T}_{l\delta} \tag{3.12}
$$

where we have used the fact that $\mathbf{H}_{N+1}$ is the same as the hypothetical $\mathbf{H}_{(\gamma_N+1)\delta}$.

Featuring Property

In fact, $\mathbf{G}$ matrices have a strong relationship with $\mathbf{F}$ matrices. We can easily verify that, for $N_b \leq l\delta < \gamma\delta \leq N$, $\mathbf{Z}_{\gamma\delta}\mathbf{G}_{\gamma\delta,l\delta}=\mathbf{F}_{\gamma\delta,l\delta}$. This is called the featuring property of $\mathbf{G}$.

The featuring property implies that, like $\mathbf{F}$ matrices, $\mathbf{Z}_{\gamma\delta}\mathbf{G}_{\gamma\delta,l\delta}$ also has the weighting property and the conformity property.

3.2.4 Rewriting Predictions

Since the prediction expression during the training phase has been provided in (3.4), we focus on the prediction expression during the predicting phase now. According to (1.7), we

get wanted predictions as

$$\mathbf{y} = \mathbf{H}_{N+1}^{\top} z\left(\mathbf{x}\right) = \sum_{l=\gamma_0}^{\gamma_N} \mathbf{T}_{l\delta}^{\top} \left(\mathbf{G}_{(\gamma_N+1)\delta,l\delta}^{\top} z\left(\mathbf{x}\right)\right). \tag{3.13}$$

with (3.12) substituted. According to the featuring property of $\mathbf{G}$, $\left(\mathbf{G}_{(\gamma_N+1)\delta,l\delta}^{\top} z\left(\mathbf{x}\right)\right)^{\top}$ is mathematically a row of $\mathbf{F}$. Thus, elements of $\mathbf{G}_{(\gamma_N+1)\delta,l\delta}^{\top} z\left(\mathbf{x}\right)$ have the conformity property as

$$\left[\mathbf{G}_{(\gamma_N+1)\delta,l\delta}^{\top} z\left(\mathbf{x}\right)\right]_i \approx \left[\left(1-\mu\alpha\right)\left(1-a_{\gamma_N+1-l,i}\right)\right]^{\gamma_N-l} \left[\mathbf{B}_{l\delta}\right]_i$$

where

$$\left[\mathbf{B}_{l\delta}\right]_i = \frac{\mu}{N_b} z^{\top}\left(\mathbf{x}_{l\delta-N_b+i}\right) z\left(\mathbf{x}\right)$$

and $a_{\gamma_N+1-l,i}$ is in the same order as $\frac{\mu}{1-\mu\alpha} \cdot \frac{1}{N_b} \sum_{k=1}^{N_b} \frac{\xi_{\gamma,i,l,k}\xi_{l,k}}{\xi_{\gamma,i}}$ with $\xi_{l,k} = \kappa\left(\mathbf{x}_{l\delta-N_b+k}, \mathbf{x}\right)$ and $N_b \leq l\delta < \gamma\delta \leq N$. So, (3.13) can be further expressed as

$$\mathbf{y} \approx \sum_{l=\gamma_0}^{\gamma_N} \sum_{i=1}^{N_b} \left[\left(1-\mu\alpha\right)\left(1-a_{\gamma_N+1-l,i}\right)\right]^{\gamma_N-l} \frac{\mu}{N_b} \xi_{l,i} \mathbf{t}_{l\delta-N_b+i}. \tag{3.14}$$

It is seen that the prediction in the predicting phase is actually a weighted sum of graph signal observations seen in the training phase, with weights being the kernel value under different scaling. The approximation in (3.14) plays a central role in finding influences of hyper parameters in the model.

3.3 Parameter Analysis

3.3.1 N_b and δ

According to (3.14), we know that a model prediction is a weighted sum of training observations with weights being scaled kernel values. The kernel value measures how similar a training input and tested input are. The scaled factors of the kernel values are related with batch indices difference $\gamma_N - l$. With suitable learning rates and kernels, the exponential term in front of the kernel value attenuates the kernel value. Mathematically, the batch size N_b is the number of observations (part of) whose kernel values are attenuated by exponential terms with the same power. The refreshing period δ controls how often the model updates its trainable parameter. The influence of the batch size N_b and the refreshing period δ on observation weights should be considered jointly.

When $N_b > \delta$, we can infer that some observations are over-weighted than others. Take Fig. 3.1 as an example. Fig. 3.1 shows 10 training observations with $N_b = 4$ and $\delta = 3$. Note that the observations are parts of a training process. In the figure, weights of observations in Batch 1 are their kernel values without attenuation, weights of observations in Batch 2 are kernel values scaled by exponential with power of 1. Observation No. 4 is considered in both Batch No. 1 and Batch No. 2, and Observation No. 7 is considered in both Batch No. 2 and Batch No. 3. The weight of Observation No. 4 in the final prediction is the sum of its weights in Batches No. 1 and No. 2. Whereas, others, such as Observations No. 2, No. 3, No. 6, No. 7 and so on, are considered only once. That makes Observations No. 4 and No. 7 to tend to have higher weights than those considered only once. That is what we call *weighting disorder*. However, when N_b is a multiple of δ, other than the first few observations in the first few batches and the last few observations in the last few batches, the other observations are considered multiple times equally. For example, we have 1000 training observations and we set $N_b = 10$ and $\delta = 1$. Then, Observations No. 1 and No. 1000 are

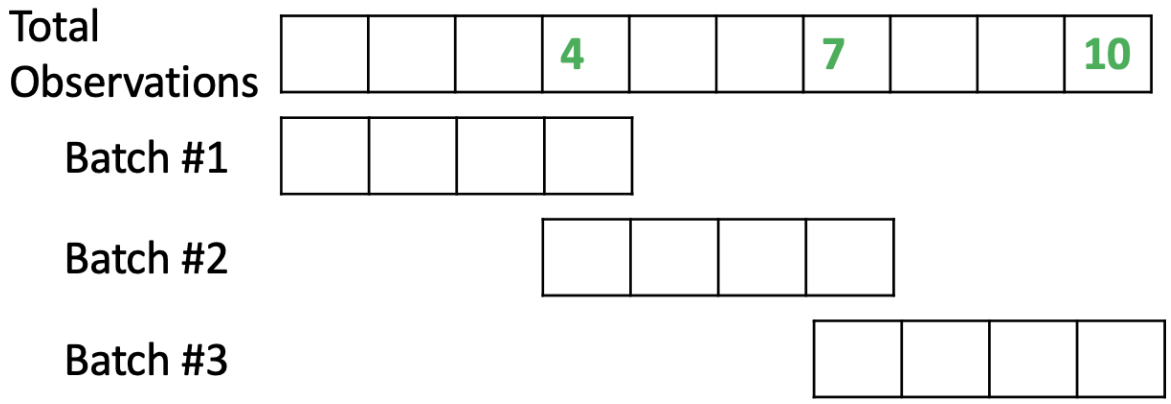

Figure 3.1: An example of $N_b = 4$ and $\delta = 3$, $\delta < N_b$, for 10 training input-output pairs.

considered once; Observations No. 2 and No. 999 are considered twice; Observations No. 3 and No. 998 are considered three times; ...; Observations No. 9 and No. 992 are considered nine times; Observations No. 10, No. 11, No. 12, ..., No. 990, and No. 991 are all considered ten times. These make the weighting disorder effect less severe.

When $N_b = \delta$, the weighting disorder effect is gone and all observations are considered only once. As a result, all training observations are taken into account based on how close their inputs and the tested input are. Although that sounds making more sense than the case of $N_b > \delta$, setting $N_b = \delta$ may not perform better than setting $N_b > \delta$. Because setting $N_b > \delta$ gives a possibility to put higher weights on more similar observations than setting $N_b = \delta$. But the possibility depends on the training sequence and setting $N_b > \delta$ leads to higher computational cost.

When $N_b < \delta$, we can foresee that some examples are actually ignored by the model.

3.3.2 Kernels

Shift-Invariant Kernels

According to [32], the kernel used in the KRG-RFF model has to be shift-invariant. Authors of [23] provide 3 shift-invariant kernels, i.e., Gaussian kernel, Laplace kernel, and Cauchy kernel. For a Gaussian kernel

$$\kappa\left(\mathbf{x}_1, \mathbf{x}_2\right) = e^{-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|_2^2}{2\sigma^2}}, \tag{3.15}$$

the random features should be drawn from $\mathcal{N}\left(\mathbf{0}, \sigma^{-\in}\mathbf{I}\right)$ and the proof can be found in [50]. The following claims show how to generate random features for a Laplace kernel and a Cauchy kernel.

Claim 3.3. *Supposing a Laplace kernel κ with a scale parameter b, i.e.,*

$$\kappa\left(\mathbf{x}_1, \mathbf{x}_2\right) = e^{-\frac{\|\mathbf{x}_1 - \mathbf{x}_2\|_1}{b}}, \tag{3.16}$$

random features $\mathbf{v}$ should be drawn such that each element of $\mathbf{v}$ are drawn from the distribution $p(v)$ independently and

$$p(v) = \frac{b}{\pi\left[1 + (bv)^2\right]}$$

when using the RFF approximation in (1.5) and (1.6).

Proof. For simplicity, let us denote the Laplace kernel by $\kappa\left(\mathbf{x}\right) = e^{-\frac{\|\mathbf{x}\|_1}{b}}$. From [23], we know random features $\mathbf{v}$ should be drawn from the distribution of $f\left(\omega\right) = \mathcal{F}\{\kappa\left(\mathbf{x}\right)\}/(2\pi)$. According to the definition of the K-dimensional Fourier transform, we get

$$f\left(\boldsymbol{\omega}\right) = \left(\frac{1}{2\pi}\right)^K \int_{\mathbb{R}^K} \kappa\left(\mathbf{x}\right) e^{-j\boldsymbol{\omega}^\top \mathbf{x}} d\mathbf{x}$$

where $j^2 = -1$. Since the l_1 norm of a vector can be written as $\|\mathbf{x}\|_1 = \sum_{k=1}^{K} |x_k|$ where x_k is the k-th element of $\mathbf{x}$, together with $\boldsymbol{\omega}^\top \mathbf{x} = \sum_{k=1}^{K} \omega_k x_k$ where ω_k is the k-th element of $\boldsymbol{\omega}$, we can express $f(\boldsymbol{\omega})$ in

$$f(\boldsymbol{\omega}) = \left(\frac{1}{2\pi}\right)^K \prod_{k=1}^{K} \int_{-\infty}^{\infty} e^{-\frac{|x_k|}{b}} e^{-j\omega_k x_k} dx_k.$$

Following the well-known Fourier transform pair $\mathcal{F}\{e^{-a|t|}\} = \frac{2a}{a^2+\omega^2}$, we have

$$f(\boldsymbol{\omega}) = \left(\frac{1}{2\pi}\right)^K \prod_{k=1}^{K} \frac{2b}{1+(b\omega_k)^2} = \prod_{k=1}^{K} \frac{b}{\pi\left(1+(b\omega_k)^2\right)}.$$

That is, each element of the random feature can be drawn independently, and identically from the distribution $p(v) = \frac{b}{\pi\left(1+(b\omega_k)^2\right)}$. ■

Claim 3.4. *Supposing a Cauchy kernel κ with a scale parameter ψ, i.e.,*

$$\kappa(\mathbf{x}_1, \mathbf{x}_2) = \prod_{k=1}^{K} \frac{1}{1+\left(\frac{x_{1,k}-x_{2,k}}{\psi}\right)^2} \tag{3.17}$$

where $x_{n,k}$ denotes the k-th element of $\mathbf{x}_n$, random features $\mathbf{v}$ should be drawn such that each element of $\mathbf{v}$ are drawn from the distribution $p(v)$ independently and

$$p(v) = \frac{\psi}{2} e^{-\psi|v|}$$

when using the RFF approximation in (1.5) and (1.6).

Proof. For simplicity, let us denote the Cauchy kernel by $\kappa(\mathbf{x}) = \prod_{k=1}^{K} \frac{1}{1+(x_k/\psi)^2}$. Similar to the proof of **Claim 3.4**, we know that the corresponding random features should be drawn from the distribution $f(\boldsymbol{\omega})$ and

$$f(\boldsymbol{\omega}) = \left(\frac{1}{2\pi}\right)^K \prod_{k=1}^{K} \int_{-\infty}^{\infty} \frac{1}{1+(x_k/\psi)^2} e^{-j\omega_k x_k} dx_k.$$

Using the duality of the Fourier transform on the well-known pair $\mathcal{F}\{e^{-a|t|}\} = \frac{2a}{a^2+\omega^2}$, we get $\mathcal{F}\{\frac{2a}{a^2+t^2}\} = 2\pi e^{-a|-\omega|}$ and

$$f(\boldsymbol{\omega}) = \prod_{k=1}^{K} \frac{\psi}{2} e^{-\psi|\omega_k|}.$$

That is, each element of the random feature can be drawn independently, and identically from the distribution $p(v) = \frac{\psi}{2} e^{-\psi|\omega_k|}$. ■

Finding a Suitable Kernel Parameter

As stated in [51], the kernel is viewed as a similarity measure between the training inputs and the current input. That expression is consistent with what we get for the KRG-RFF model. We know from (3.14) that the (scaled) kernel values are used as weights of observations. Then, it is natural to set higher weights, i.e., to have higher kernel values, for more similar inputs.

A detailed parameter analysis for Gaussian kernels in another learning model called Gradraker is provided in Section 2.4 [50] together with a method of finding a suitable Gaussian variance for the kernel. The method can be applied to the KRG-RFF model, guiding on how to find a suitable Gaussian variance. The essence of the method is to let the maximum possible kernel values to be close to 1 and the minimum possible kernel values to be close to 0 given the training dataset. So, although the method in Section 2.4 applies on Gaussian kernels only, as stated in Section 2.6, it can be easily extended to include Laplace kernels. Specifically, according to the expression of a Gaussian kernel in (3.15), the maximum possible kernel value is attained at the largest l_2-norm of the difference among all used training pairs in the given dataset, i.e., $\max_{m,n} \|\mathbf{x}_m - \mathbf{x}_n\|_2^2$. So, according to Section 2.4, we have for a Gaussian

kernel, a suitable Gaussian variance σ^2 should be such that

$$e^{-\frac{\max\limits_{m,n} \|\mathbf{x}_m - \mathbf{x}_n\|_2^2}{2\sigma^2}} = \frac{1}{\sqrt{2D}}$$

or equivalently,

$$\sigma^2 = \frac{\max\limits_{m,n} \|\mathbf{x}_m - \mathbf{x}_n\|_2^2}{\ln(2D)} \tag{3.18}$$

with $\mathbf{x}_m$ and $\mathbf{x}_n$ being different training inputs. Similarly, according to (3.16) for a Laplace kernel, the maximum possible kernel value is attained at the largest l_1-norm of difference among all used training inputs in the given dataset. So, a suitable Laplace kernel parameter σ should be such that

$$e^{-\frac{\max\limits_{m,n} \|\mathbf{x}_m - \mathbf{x}_n\|_1}{\sigma}} = \frac{1}{\sqrt{2D}}$$

or equivalently,

$$\sigma = \frac{2\max\limits_{m,n} \|\mathbf{x}_m - \mathbf{x}_n\|_1}{\ln(2D)} \tag{3.19}$$

with $\mathbf{x}_m$ and $\mathbf{x}_n$ being different training inputs. The extended method of finding a suitable parameter for Gaussian kernels and Laplace kernels based on Algorithm 2 is summarized in Algorithm 4.

Although the idea of finding maximum possible value and then setting it close to 1 can be applied on kernels such as Cauchy kernels, it is hard to find a closed-form expression for a suitable parameter for such a kernel. Because the power of the unknown variable., i.e., the wanted kernel parameter, is too high in the resulting equality. Besides, Algorithm 4 does not consider the input distribution. Therefore, we propose Algorithm 5. It considers

Algorithm 4 Extended Method: Finding Maximum Possible Value

Input: Kernel type (Gaussian or Laplace), all used training inputs
if Gaussian kernel **then**
  find $\max_{m,n} \|\mathbf{x}_m - \mathbf{x}_n\|_2^2$ among all inputs;
  get σ^2 via (3.18);
else if Laplace kernel **then**
  find $\max_{m,n} \|\mathbf{x}_m - \mathbf{x}_n\|_1$ among all inputs;
  get σ via (3.19);
end if

the distribution of used inputs and tries to find the the kernel parameter that achieves the maximum variance for the resulting kernel values for used inputs. More specifically, given a kernel parameter, we can find a kernel value for each pair of used inputs and thus a variance for the resulting kernel values. Algorithm 5 is to maximize the variance for kernel values. We have seen that kernel values for similar inputs should be larger than kernel values for dissimilar inputs. Consequently, finding a kernel parameter which achieves maximum variance of resulting kernel values is a reasonable way. In addition, Algorithm 5 can be applied on kernels such as Cauchy kernels which do not have a closed-form expression for a suitable kernel parameter.

3.3.3 Learning Rates and α

Given a kernel, different learning rates still lead to different model performance. In order to find a suitable learning rate for a model, we need to first understand how the learning rate affects the model performance. Intuitively, a suitably large learning rate helps the model converge to its optimum fast, whereas a smaller learning rate lets the model fail to accumulate enough change to achieve its optimum. That intuition can be explained by (3.14) via sum of weights of training observations $\mathbf{t}$.

Algorithm 5 Finding Maximum Kernel Variance

Input: Kernel type, all used training inputs, stopping threshold ϵ
Initialization:
set *middle* to be the result from Algorithm 4 or a random positive value and get kernel values for all input pairs;
get *varm* as the variance of acquired kernel values based on *middle*;
set $left$ to be $middle/2$ and get kernel values for all input pairs;
get *varl* as the variance of acquired kernel values based on $left$;
set *right* to be $middle * 2$ and get kernel values for all input pairs;
get *varr* as the variance of acquired kernel values based on *right*;
while $\max\{varl, varm, varr\} > \epsilon$ **do**
  if $varl < varm < varr$ **then**
    set $left$ to be *middle* and *middle* to be *right*, and update *varl* and *varm* accordingly;

    set *right* to be $right * 2$ and update *varr* accordingly;
  else if $varl > varm > varr$ **then**
    set *right* to be *middle* and *middle* to be $left$, and update *varr* and *varm* accordingly;

    set $left$ to be $left/2$ and update *varl* accordingly;
  else
    set $left$ to be $\sqrt{left \cdot middle}$ and *right* to be $\sqrt{right \cdot middle}$, and update *varl* and *varr* accordingly;
  end if
end while

Sum of Observation Weights

Let us first examine the sum of observation weights. The exact sum depends on the sequence of the training observations, the RF realization, and other data-specific factors. So, we focus on the expectation of the sum. Noticing that $a_{\gamma_N+1-l,i}$'s in (3.14) are at the same order for different l and i, we can find a substitution a also in the same order and have the following equality

$$S = \mathbb{E}\left[\sum_{l=\gamma_0}^{\gamma_N}\sum_{i=1}^{N_b}\left[(1-\mu\alpha)(1-a)\right]^{\gamma_N-l}\frac{\mu}{N_b}\xi_{l,i}\right] = \sum_{l=\gamma_0}^{\gamma_N}\sum_{i=1}^{N_b}\left[(1-\mu\alpha)(1-a)\right]^{\gamma_N-l}\frac{\mu}{N_b}\mathbb{E}\left[\xi_{l,i}\right] \tag{3.20}$$

holds. Recalling that $\xi_{l,i} = \kappa\left(\mathbf{x}_{l\delta-N_b+k}, \mathbf{x}\right)$, when the distribution of inputs is independent of time instances, the expectation $\mathbb{E}\left[\xi_{l,i}\right]$ is not related with l and i. Thus, we can have $\xi = \mathbb{E}\left[\xi_{l,i}\right]$ further reduce (3.20) as

$$S = \frac{\mu\xi \cdot \left[1 - \left[(1-\mu\alpha)(1-a)\right]^{\gamma_N - \gamma_0 + 2}\right]}{1 - (1-\mu\alpha)(1-a)}. \tag{3.21}$$

Noticing that a is in the same order as $\frac{\mu}{1-\mu\alpha} \cdot \frac{1}{N_b} \sum_{k=1}^{N_b} \frac{\xi_{\gamma,i,l,k}\xi_{l,k}}{\xi_{\gamma,i}}$, we can have the approximation $a \approx \frac{\mu\xi}{1-\mu\alpha}$ and rewrite (3.21) as

$$S \approx \frac{\xi}{\xi + \alpha}\left[1 - \left[1 - \mu(\xi+\alpha)\right]^{\gamma_N - \gamma_0 + 2}\right]. \tag{3.22}$$

The approximation (3.22) plays a central role in understanding the influence of learning rates.

Influence of μ

Considering a stationary output distribution, it makes sense that we want the expectation of the sum over observation weights S to be 1. Consequently, α has to be a small number relative to ξ such that $\frac{\xi}{\xi+\alpha} \approx 1$ holds. That is a valid requirement because we do not want the second term in (1.8) to be dominant. But $\left[1 - \left[1 - \mu(\xi+\alpha)\right]^{\gamma_N - \gamma_0 + 2}\right]$ in (3.22) is not always close to 1 because $\gamma_N - \gamma_0 + 2$ is limited. This is when μ plays a role. The influence of μ is shown in Fig. 3.2. Note that, in the figure, lower NMSE values means better performance. The possible range $(0, \infty)$ of μ is divided into three ranges, i.e., the immature range, the weighting range, and the diverging range, based on the behavior of S. The threshold of the immature range and the weighting range is denoted by μ_{iw}, and the threshold of the weighting range and the diverging range is denoted by μ_{wd}.

In the immature range, μ is so small that $\mu(\xi+\alpha) \ll 1$. As a result, we would have

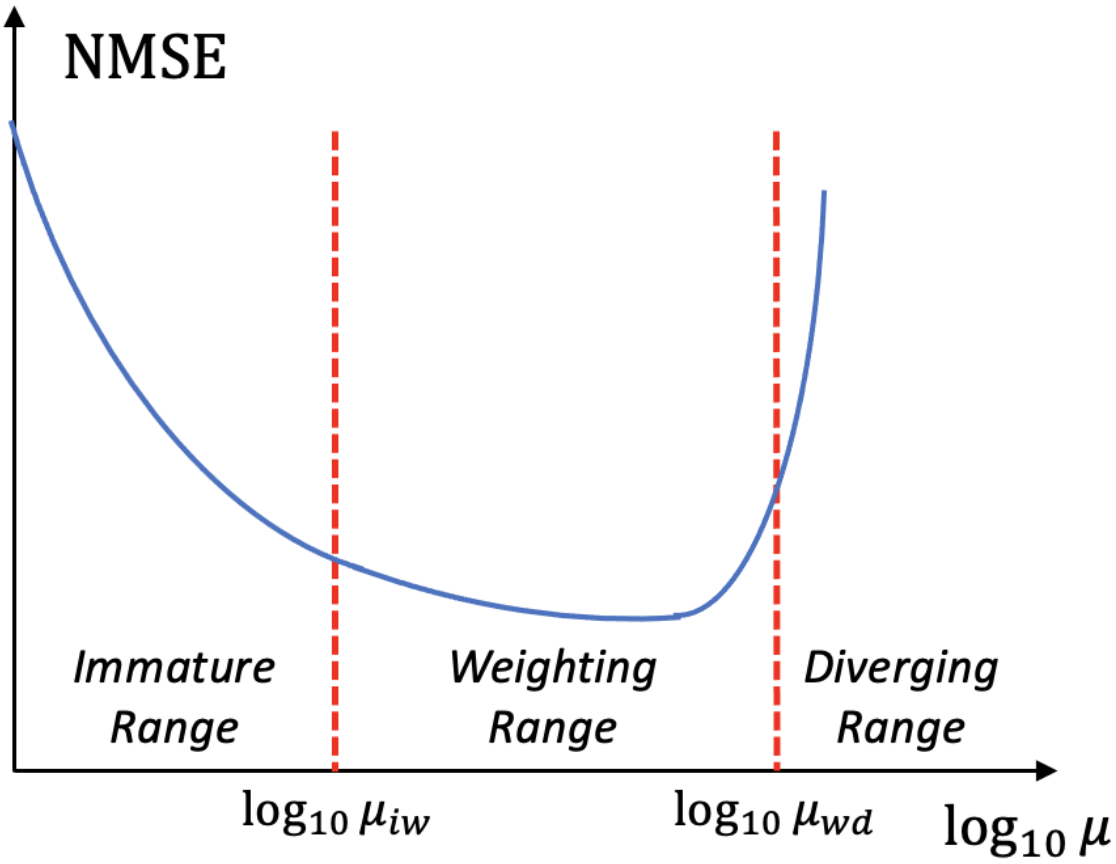

Figure 3.2: A general normalized mean squared error (NMSE) curve versus the learning rate μ given a kernel.

$1-\mu(\xi+\alpha) \approx 1$ and $S \ll 1$. That means observations do not gain large enough weights to compose a model prediction with a similar distribution to the distribution of observations. The model performance is bad. As μ increases in the immature range, S becomes larger and the model performance gets better.

Increasing μ till $S \approx 1$, then we are looking at the weighting range. In the weighting range, μ is suitable such that $\mu(\xi+\alpha) \approx 1$. We would like to mention that when $\mu(\xi+\alpha) < 1$, the weights of observations are generally positive. When $\mu(\xi+\alpha) > 1$, the weights of observations would alter between positive and negative based on the batch indices they belong to. Notice that both of the situations could attain $S \approx 1$. When the negative weights are not dominant,

the model performance is acceptable. So, generally speaking, increasing μ slightly improves the model performance while all of them achieve acceptable performance.

Continuing increasing μ from the weighting range, we will finally look at the diverging range. In this range, μ is too large such that $1-\mu(\xi+\alpha) < -1$ and the absolute value of S becomes far larger than 1. The model performance is bad and increasing μ further brings worse performance.

As a result, we choose μ such that

$$\mu(\xi + \alpha) = 1$$

or equivalently

$$\mu = \frac{1}{\xi + \alpha} \tag{3.23}$$

as a suitable learning rate.

3.4 Simulation Results

We provide simulation results about the influence of parameters in the KRG-RFF model on observation weights and thus model performance using real datasets. Specifically, we show behavior of the weights using different combinations of the batch size N_b and the refreshing period δ, the performances of the two methods, i.e., Algorithm 4 and Algorithm 5, on finding a suitable kernel parameter, behavior of the weights under different learning rates μ given a kernel, and the performance of the proposed suitable μ.

Following [32], we use the normalized mean squared error in dB, NMSE (dB), as the perfor-

mance metric

$$\text{NMSE} = 10\log_{10}\left(\mathbb{E}\left[\frac{\|\mathbf{Y}-\mathbf{T}\|_F^2}{\|\mathbf{T}\|_F^2}\right]\right) \tag{3.24}$$

for tested data.

3.4.1 Real Datasets

Norway Temperature

The data is collected by the Norwegian Meteorological Institute [52]. The dataset used in the paper contains daily average temperature of $K = 36$ Norway meteorological stations during January 1st, 2020 to December 31st, 2020.

The weighted, undirected graph is constructed using the information for January 1st, 2020. First, we create a directed 5-nearest-neighbor (5NN) graph, based on which we create a weighted undirected graph where there is an edge if there exists at least one directed edge between the two nodes in the 5NN graph. The undirected edge weight is proportional to the absolute difference between the two nodal values. More specifically, if there exists an undirected edge between nodes v_i and v_j, then the edge weight is

$$w_{i,j} = w_{j,i} = \frac{|\,[\mathbf{y}]_i - [\mathbf{y}]_j\,|}{\max(\mathbf{y}) - \min(\mathbf{y})} \tag{3.25}$$

where $\mathbf{y}$ is a graph signal and $[\mathbf{y}]_i$ is the corresponding nodal value for v_i.

US Temperature

The US Temperature dataset used in the paper is from the Average Daily Temperature Archive [53]. Source data of the site are from the National Climatic Data Center. The site contains the daily temperature of more than 300 cities all over the world from January 1st, 1995 to present. The US Temperature dataset consists of daily temperature for $K = 143$ cities in the US from January 1st, 2019 to December 31st, 2019.

Similar to the graph in the Norway Temperature, the weighted, undirected graph in the US Temperature dataset is generated based on the directed nearest-neighbor (8NN this time) using the information for Jan. 1st, 2019 with weight in (3.25).

3.4.2 Weight Behavior Under N_b and δ

We use the Norway Temperature dataset to show influences of different combinations of the batch size N_b and the refreshing period δ. Recall that we view the daily temperature for all the cities on a day as a graph signal. My goal is to predict the daily temperature on the next day given a graph signal. We randomly shuffle the sequence of the graph signals in the dataset and choose 70% of total today-tomorrow pairs as training input-output pairs.

In the experiment, we use a Gaussian kernel with Gaussian variance $\sigma^2 = 1000$ and learning rate $\mu = 0.05$. We also set $\alpha = 0.01$, $\beta = 0.001$, the number of RF $D = 4K$. The behavior of observation weights for a tested input under different combinations of N_b and δ are shown in Fig. 3.3. In the figures, the blue lines denote the (scaled by $\frac{\mu}{N_b}$) kernel values of the training inputs at the training time instances and the tested input, i.e., $\frac{\mu}{N_b}\kappa\left(\mathbf{x}_n, \mathbf{x}\right), N_b \leq n \leq N$. The red lines denote the weights of the observations. We can see from Fig. 3.3a with $N_b = \delta$ that weights are decaying from the end of training to the beginning in general, and that large (scaled) kernel values tend to lead to large weights. In Fig. 3.3b, we can see that

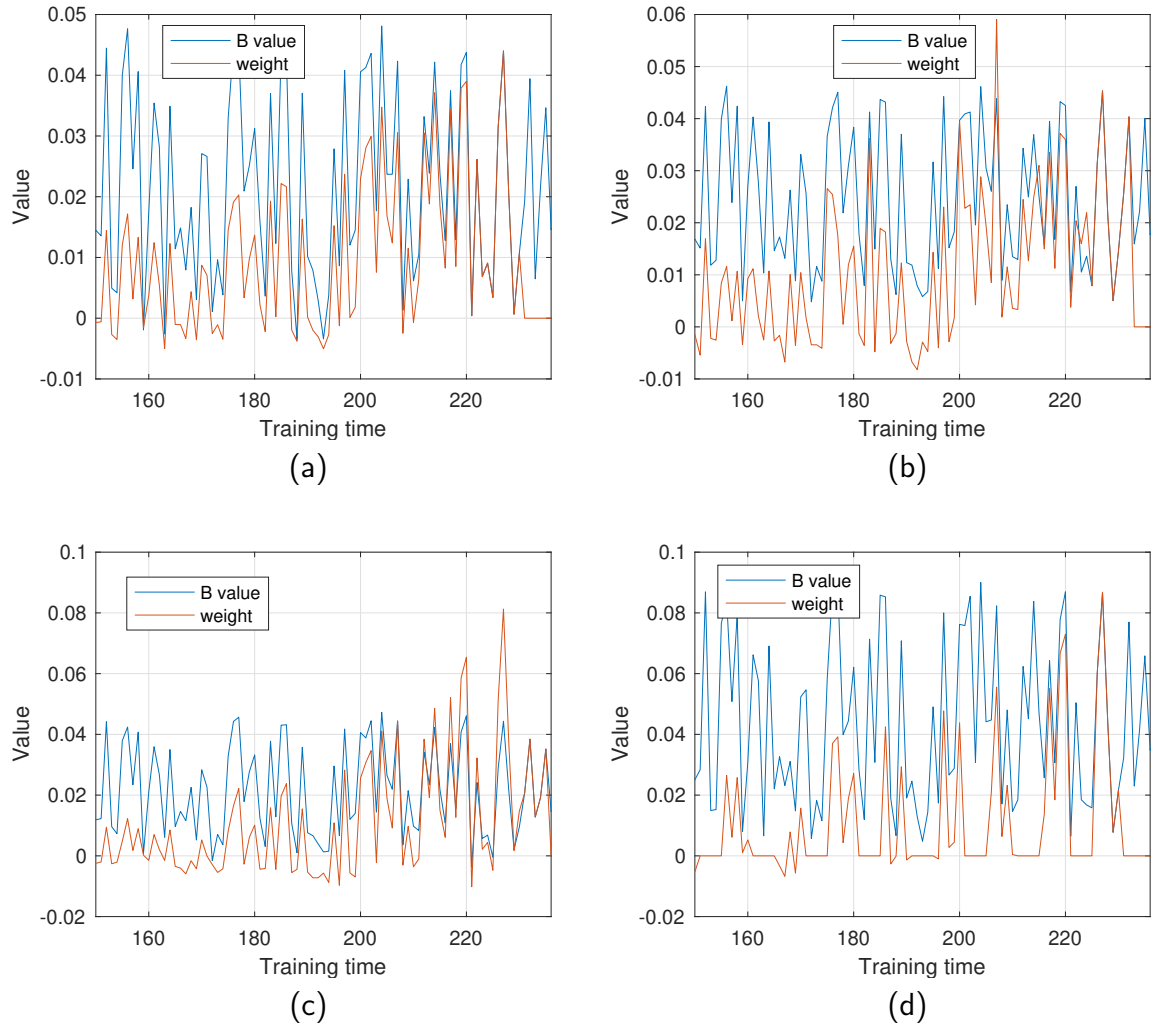

Figure 3.3: Behavior of observation weights under different combinations of the batch size N_b and the refreshing period δ. The resulting NMSE (dB) values are reported. (a) $N_b = 10$, $\delta = 10$, NMSE (dB) is -10.02. (b) $N_b = 10$, $\delta = 8$, NMSE (dB) is -10.06. (c) $N_b = 10$, $\delta = 5$, NMSE (dB) is -10.79. (d) $N_b = 5$, $\delta = 10$, NMSE (dB) is -9.86.

there is a very high weight at training time 207. The observation gains a weight higher than its (scaled) kernel value whereas other observations with similar (scaled) kernel values gain obviously smaller weights. This is an example of the weighting disorder phenomenon. This phenomenon is less severe in Fig. 3.3c where $N_b = 2\delta$. In Fig. 3.3c, although the last few examples have smaller weights, weights of other observations change exponentially in general. It is clear in Fig. 3.3d with $N_b < \delta$ that some observations have weights equal to 0.

From the perspective of NMSE (dB), there is not much difference among the situations. However, behavior of the weights better interprets what is happening in different situations which are seemingly alike.

3.4.3 Finding a Suitable Kernel Parameter

Comparing with the batch size N_b and the refreshing period δ selection, kernel parameter selection has a huge influence on model performance. In fact, kernel parameter selection determines the lowest achievable NMSE (dB) value for the model in an application. We proposed two methods in Section 3.3.2 for finding a suitable kernel parameter. The performance of the two methods are reported in Table 3.1 and Table 3.2. In both tables, the first numerical column represents the lowest NMSE (dB) values achievable with the kernel parameter found by Algorithm 4. The second numerical column represents the lowest NMSE (dB) values with the kernel parameter found by Algorithm 5. The third numerical column represents the lowest NMSE (dB) values found by trial and error. The difference between Table 3.1 and Table 3.2 is that Table 3.1 uses graph signals in their original sequence whereas Table 3.2 uses randomly shuffled graph signals, so that we would have different input distributions for the two groups of simulations. We set $N_b = \delta = 10$, $\alpha = 0.01$, $\beta = 0.001$, and the number of RFs $D = 4K$. Reported numbers are averages over 50 repeated experiments.

Table 3.1: Lowest NMSE (dB) values found by the two methods and by simulation using different kernels and the two datasets. The graph signals sequence is in order.

		Alg. 1	**Alg. 2**	Sim.
Norway	Gaussian	-4.39	**-6.22**	**-6.22**
	Laplace	-4.91	**-5.4**	**-5.87**
	Cauchy	-	**-6.13**	**-6.21**
US	Gaussian	-14.01	**-15.5**	**-15.5**
	Laplace	**-15.4**	**-15.4**	**-15.7**
	Cauchy	-	**-15.65**	**-15.65**

Table 3.2: Lowest NMSE (dB) values found by the two methods and by simulation using different kernels and the two datasets. The graph signals sequence is shuffled.

		Alg. 1	**Alg. 2**	Sim.
Norway	Gaussian	-10.36	**-10.48**	**-10.48**
	Laplace	**-10.37**	**-10.37**	**-10.37**
	Cauchy	-	**-10.44**	**-10.44**
US	Gaussian	**-16.8**	-16.2	**-16.85**
	Laplace	**-17.43**	**-17.43**	**-17.5**
	Cauchy	-	**-16.6**	**-17.04**

In Table 3.1 and Table 3.2, the values in the third numerical column are highlighted since that is the best one can achieve in the specific application with a specific kernel. The number(s) in the first two column of the same row is (are) highlighted as well. We can see that the performance of Algorithm 5 is slightly better than Algorithm 4, but both of the methods could achieve acceptable performance.

3.4.4 Weight Behavior Under μ

While a suitable kernel parameter gives the possibility to achieve low NMSE (dB) values, the model would achieve its optimum only when the learning rate μ is also chosen properly. We first show the influence of μ on observation weights behavior, and then show the performance of the proposed formula (3.23) of finding a suitable μ.

Fig. 3.4 shows observation weights behavior under different learning rates using the Norway Temperature Dataset. In the experiments, $N_b = \delta = 10$, $\alpha = 0.01$, $\beta = 0.001$, and $D = 4K$. We use a Gaussian kernel with Gaussian variance found by Algorithm 5. The resulting kernel variance is $\sigma^2 = 0.43$. When we have $\mu = 0.4$, we would get $S \approx 0.97$ according to (3.22). So, we set $\mu_{iw} = 0.4$. In addition, we have $\mu_{wd} = \frac{2}{\xi+\alpha} = 4.55$.

Fig. 3.4a shows for the case $\mu = 0.03$. In the figure, the observation weights are generally small and the actual sum of weights is 0.26. The model performance is bad in terms of

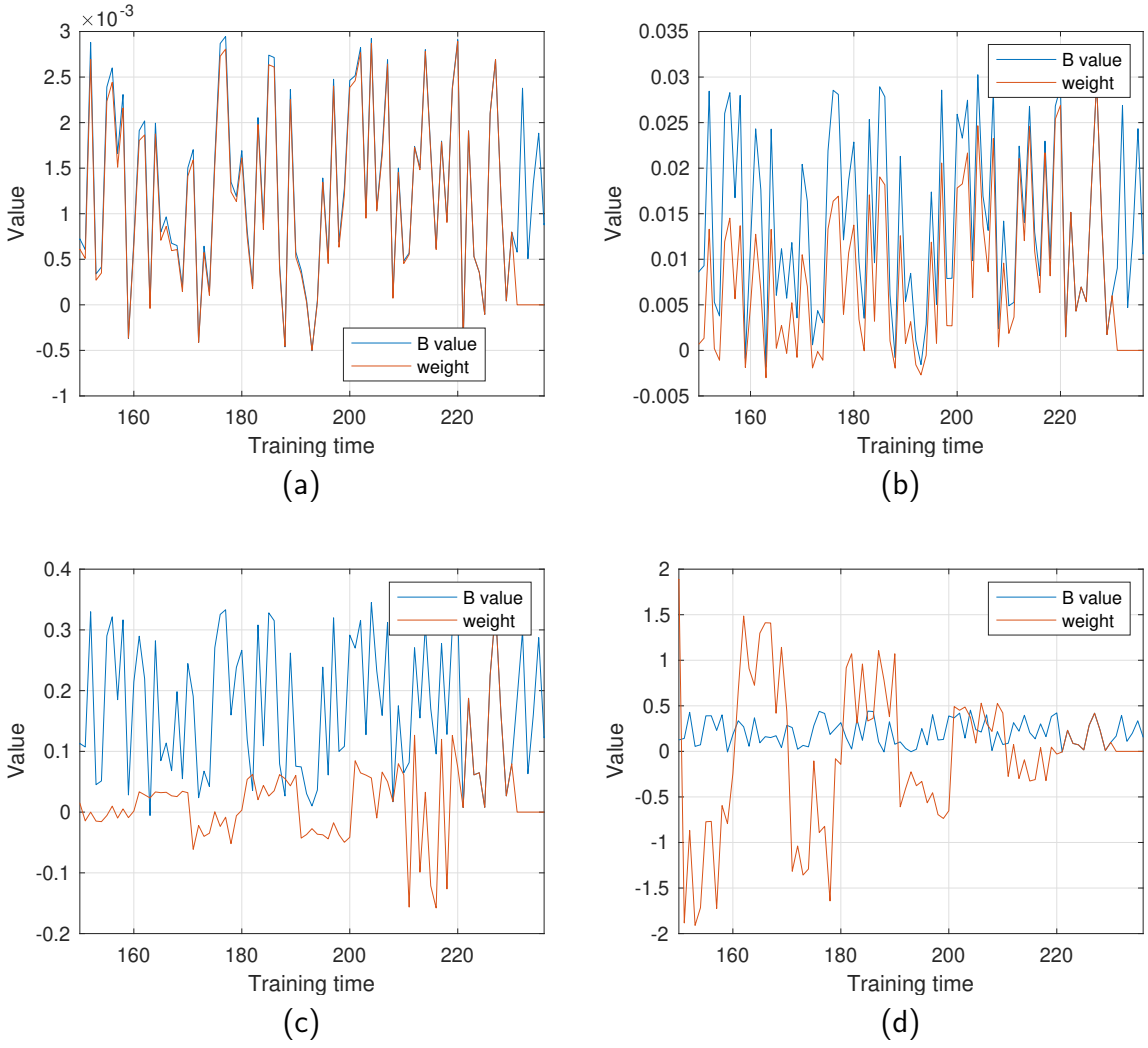

Figure 3.4: Observation weights behavior under different learning rates. The sums of weights and the resulting NMSE (dB) values are reported. (a) $\mu = 0.03$, $S = 0.26$, NMSE (dB) is -1.54. (b) $\mu = 0.3$, $S = 0.98$, NMSE (dB) is -8.6. (c) $\mu = 3.6$, $S = 2.53$, NMSE (dB) is -8.22. (d) $\mu = 5.2$, $S = -220.46$, NMSE (dB) is 61.22.

NMSE (dB) value. As μ approaches μ_{iw}, for example, $\mu = 0.3$ as shown in Fig. 3.4b, the actual sum of weights comes to 0.98 and NMSE (dB) value is decreased to -8.6. When we continue to increase μ but keep μ to stay within the weighting range, for example, making $\mu = 3.6$, the actual sum of weights is increased to 2.53 but the NMSE (dB) stays roughly the same. However, when we increase μ till reaching the diverging range, i.e., larger than μ_{wd}, for example, setting $\mu = 5.2$, the absolute value of the actual sum of weights is more than 200, and the NMSE (dB) value is up to 61.22.

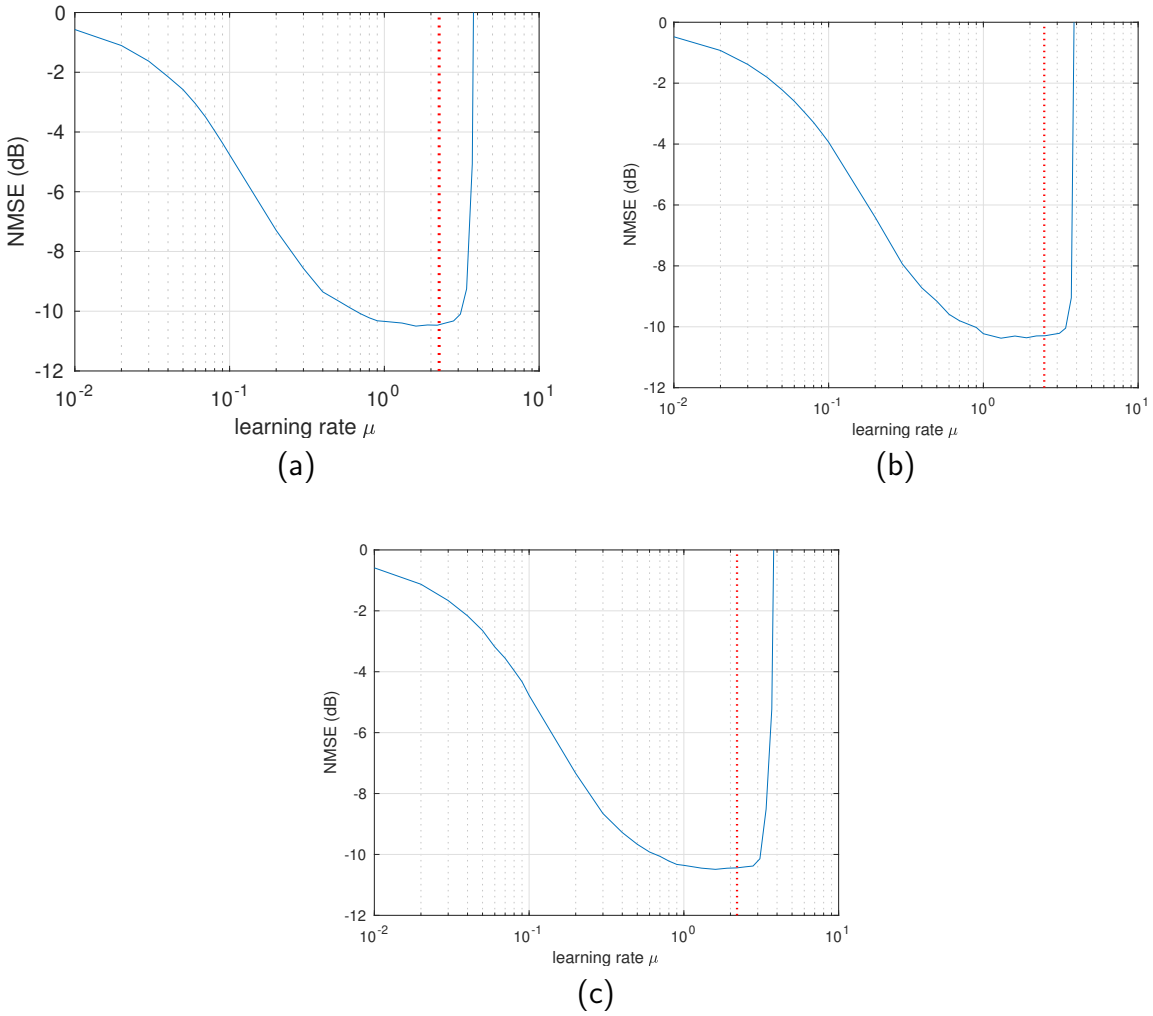

Figure 3.5: NMSE (dB) found by simulation with different learning rates and the position of the proposed learning rate via (3.23) for different kernels using the Norway Temperature dataset. (a) a Gaussian kernel with $\sigma^2 = 948$. (b) a Laplace kernel with $b = 221$. (c) a Cauchy kernel with $\psi = 44$.

We use the Norway Temperature dataset to check how well the proposed μ in (3.23) performs. The results are shown in Fig. 3.5 for the three shift-invariant kernels. In the experiments, we set $N_b = \delta = 10$, $\alpha = 0.01$, $\beta = 0.001$, and $D = 4K$. Kernel parameters are found via Algorithm 5. Blue curves denote NMSE (dB) values found in simulation with corresponding learning rates. Note that each point on blue curves are averages over 50 repeated experiments. The red dotted lines denote theoretical μ values via (3.23). We can see that (3.23) finds suitable learning rates for different kernels.

Chapter 4

Conclusion

This dissertation presented a parameter analysis framework for kernel-based regression models over graphs with random Fourier feature approximation and batch-based training solutions. The main idea of the framework is to express the model prediction, either a scalar or a vector, with respect to training observations. According to the framework, we find that the weights of observations when calculating a model prediction are scaled kernel values between the observation inputs and the tested input. The influences of hyper-parameters, such as kernel parameters and learning rates, on model predictions are understood by figuring out the influences of hyper parameters on observation weights.

The recursion nature of the training process is shown in the definition of observation weights. We put effort on figuring out properties of the observation weights, especially being interested in understanding how those weights are related with different hyper parameters in the model. Although acquiring exact properties is hard due to the recursion nature, we managed to get insights through approximations. The weights of observations are approximately kernel values multiplied by an exponential factor whose base is related with the kernel value average over the input difference distribution and the exponent is related with the time

difference between the two inputs for the kernel. The influence of learning rates are found by checking summations of weights of observations. The influence of the batch size and the refreshing period, if there are such two parameters, should be considered jointly. Based on such understandings on parameters, methods of finding suitable model parameters are proposed.

KRG-RFF models are confirmed to have good performance and iterative learning makes KRG-RFF application on large networks possible. Using the proposed model parameters in the dissertation avoids wasting physical and temporal resources on tuning the model to its optimal status in an application, and at the mean time, makes the model more interpretable.

Bibliography

[1] R. J. Wilson. *Introduction to graph theory (4th edition)*, chapter 1,2. Addison Wesley Longman Limited, 1996.

[2] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. Collective classification in network data. *AI Magazine*, 29(3):93–106, 2008.

[3] Y. Liu, L. Yang, K. You, W. Guo, and W. Wang. Graph learning based on spatiotemporal smoothness for time-varying graph signal. *IEEE Access*, 7:62372–62386, 2019.

[4] J. Wu, F. Orlandi, D. O'Sullivan, and S. Dev. Publishing climate data as linked data via virtual knowledge graphs. In *IGARSS 2022 - 2022 IEEE International Geoscience and Remote Sensing Symposium*, pages 4090–4093, 2022.

[5] Y. Li and G. Mateos. Graph frequency analysis of covid-19 incidence to identify county-level contagion patterns in the united states. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3230–3234, 2021.

[6] J. Giraldo, A. Mahmood, B. Garcia-Garcia, D. Thanou, and T. Bouwmans. Reconstruction of time-varying graph signals via sobolev smoothness. *IEEE Transactions on Signal and Information Processing over Networks*, 8:201–214, 2022.

[7] G. Li, V. Knoop, and H. Lint. Dynamic graph filters networks: A gray-box model for multistep traffic forecasting. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020.

[8] M. Petrovic, R. Liegeois, T. Bolton, and D. Van De Ville. Community-aware graph signal processing: Modularity defines new ways of processing graph signals. *IEEE Signal Processing Magazine*, 37(6):150–159, 2020.

[9] R. Ramakrishna and A. Scaglione. Grid-graph signal processing (grid-gsp): A graph signal processing framework for the power grid. *IEEE Transactions on Signal Processing*, 69:2725–2739, 2021.

[10] W. Huang, T. Bolton, J. Medaglia, D. Bassett, A. Ribeiro, and D. Van De Ville. A graph signal processing perspective on functional brain imaging. *Proceedings of the IEEE*, 106(5):868–885, 2018.

[11] A. Sandryhaila and J. M. F. Moura. Discrete signal processing on graphs. *IEEE Trans. Signal Process.*, 61(7):1644–1656, Apr. 2013.

[12] A. Sandryhaila and J. M. F. Moura. Discrete signal processing on graphs: Frequency analysis. *IEEE Trans. Signal Process.*, 62(12):3042–3054, Jun. 2014.

[13] 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.*, 30(3):83–98, May 2013.

[14] A. Ortega, P. Frossard, J. Kovačević, J. M. F. Moura, and P. Vandergheynst. Graph signal processing: Overview, challenges, and applications. *Proceedings of the IEEE*, 106(5):808–828, May 2018.

[15] A. G. Marques, S. Segarra, G. Leus, and A. Ribeiro. Sampling of graph signals with successive local aggregations. *IEEE Trans. Signal Process.*, 64(7):1832–1843, Apr. 2016.

[16] X. Wang, P. Liu, and Y. Gu. Local-set-based graph signal reconstruction. *IEEE Trans. Signal Process.*, 63(9):2432–2444, May 2015.

[17] S. K. Narang, A. Gadde, and A. Ortega. Signal processing techniques for interpolation in graph structured data. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 5445–5449, 2013.

[18] A. Sandryhaila and J. M. F. Moura. Discrete signal processing on graphs: Graph filters. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 6163–6166, 2013.

[19] Z. Xiao, H. Fang, and X. Wang. Distributed nonlinear polynomial graph filter and its output graph spectrum: Filter analysis and design. *IEEE Trans. Signal Process.*, 69:1725–1739, Mar. 2021.

[20] R. I. Kondor and J. D. Lafferty. Diffusion kernels on graphs and other discrete input spaces. *Proceedings of the Nineteenth International Conference on Machine Learning*, page 315–322, Jul. 2002.

[21] A. J. Smola and R. Kondor. Kernels and regularization on graphs. *Learning Theory and Kernel Machines*, pages 144–158, 2003.

[22] D. Romero, M. Ma, and G. B. Giannakis. Kernel-based reconstruction of graph signals. *IEEE Trans. Signal Process.*, 65(3):764–778, Feb. 2017.

[23] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proc. Advances in Neural Info. Process. Syst.*, pages 1177–1184, Vancouver, Canada, Dec. 2007.

[24] X. Dong, D. Thanou, P. Frossard, and P. Vandergheynst. Learning Laplacian matrix in smooth graph signal representations. *IEEE Trans. Signal Process.*, 64(23):6160–6173, Dec 2016.

[25] M. Coutino, E. Isufi, and G. Leus. Advances in distributed graph filtering. *IEEE Transactions on Signal Processing*, 67(9):2320–2333, 2019.

[26] S. Fortunato and D. Hric. Community detection in networks: A user guide. *Physics Reports*, 659:1–44, nov 2016.

[27] V. Paulsen and M. Raghupathi. *An Introduction to the Theory of Reproducing Kernel Hilbert Spaces*, chapter 1,2. Cambridge Studies in Advanced Mathematics. Cambridge University Press, 2016.

[28] B. Schölkopf, R. Herbrich, and A. Smola. A generalized representer theorem. In David Helmbold and Bob Williamson, editors, *Computational Learning Theory*, pages 416–426, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[29] Kernel method. `https://en.wikipedia.org/wiki/Kernel_method`. Accessed: 2022-11-09.

[30] A. Venkitaraman, S. Chatterjee, and P. Händel. Predicting graph signals using kernel regression where the input signal is agnostic to a graph. *IEEE Transactions on Signal and Information Processing over Networks*, 5(4):698–710, August 2019.

[31] Kronecker product. `https://en.wikipedia.org/wiki/Kronecker_product`. Accessed: 2023-01-09.

[32] V. Elias, V. Gogineni, W. Matins, and S. Werner. Kernel regression over graphs using random Fourier features. *IEEE Transactions on Signal Processing*, 70:936–949, March 2022.

[33] Y. Shen, G. Leus, and G. B. Giannakis. Online graph-adaptive learning with scalability and privacy. *IEEE Transactions on Signal Processing*, 67(9):2471–2483, May 2019.

[34] V. Vovk. A game of prediction with expert advice. *Journal of Computer and System Sciences*, 56(2):153–173, 1998.

[35] V. N. Ioannidis, P. A. Traganitis, Y. Shen, and G. B. Giannakis. Kernel-based semi-supervised learning over multilayer graphs. *Proc. IEEE Int. Workshop Signal Process. Advances Wireless Commun.*, pages 1–5, June 2018.

[36] Federal Office of Meteorology and Climatology MeteoSwiss. Climate normals 1961-1990: Air temperature 2m. Technical report, 2020.

[37] Federal Office of Meteorology and Climatology MeteoSwiss. Climate normals 1981-2010: Air temperature 2m. Technical report, 2020.

[38] Z. Zong and Y. Shen. Online multi-hop information based kernel learning over graphs. *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2980–2984, 2021.

[39] Y. Shen, T. Chen, and G. B. Giannakis. Online ensemble multi-kernel learning adaptive to non-stationary and adversarial environments. In *Proc. of Intl. Conf. on Artificial Intelligence and Statistics*, Lanzarote, Canary Islands, Apr. 2018.

[40] F. Chollet. *Deep learning with Python*. Manning, 2018.

[41] V. R. M. Elias, V. C. Gogineni, W. A. Martins, and S. Werner. Adaptive graph filters in reproducing kernel Hilbert spaces: Design and performance analysis. *IEEE Trans. Signal Process.*, 7:62–74, 2021.

[42] B. Osgood. The Fourier transform and its applications. lecture notes for EE 261. [Online]. Available: https://see.stanford.edu/materials/lsoftaee261/book-fall-07.pdf.

[43] J. G. Proakis. *Digital Communications (4th Edition)*, page 35. McGraw-Hill, Boston, 2000.

[44] J. Leskovec, J. Kleinberg, and C. Faloutsos. Graph evolution: Densification and shrinking diameters. *ACM Trans. on Knowledge Discovery from Data*, 1(1):1–41, March 2007.

[45] B. Rozemberczki, P. Scherer, Y. He, G. Panagopoulos, M. Astefanoaei, O. Kiss, F. Beres, N. Collignon, and R. Sarkar. PyTorch Geometric Temporal: Spatiotemporal Signal Processing with Neural Machine Learning Models, 2021.

[46] E. Isufi, A. Loukas, N. Perraudin, and G. Leus. Forcasting time series with VARMA recursions on graphs. *IEEE Trans. Signal Process.*, 67(18):4870–4885, Sep. 2019.

[47] F. Grassi, A. Loukas, N. Perraudin, and B. Ricaud. A time-vertex signal processing framework: Scalable processing and meaningful representations for time-series on graphs. *IEEE Trans. Signal Process.*, 66(3):817–829, Feb. 2018.

[48] A. Loukas, E. Isufi, and N. Perraudin. Predicting the evolution of stationary graph signals. In *2017 51st Asilomar Conference on Signals, Systems, and Computers*, pages 60–64, 2017.

[49] V. Elias, V. Gogineni, W. Martins, and S. Werner. Adaptive graph filters in reproducing kernel hilbert spaces: Design and performance analysis. *IEEE Transactions on Signal and Information Processing over Networks*, 7:62–74, 2021.

[50] Y. Zhao and E. Ayanoglu. Gaussian kernel variance for an adaptive learning method on signals over graphs. *IEEE Transactions on Signal and Information Processing over Networks*, 8:389–403, 2022.

[51] T. Hofmann, B. Schölkopf, and A. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171–1220, 2008.

[52] Norsk Klimaservicesenter, eKlima, Norway, 2022. `https://seklima.met.no/`. Accessed: 2022-08-04.

[53] University of Dayton. Environmental protection agency average daily temperature archive. `http://academic.udayton.edu/kissock/http/Weather/default.htm`. Accessed: 2022-11-21.