Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Efficient approximation of random basis functions for covariance kernels

Sheng-Chia Yu

Computational Science and Engineering
(International Master's Program)

Technische Universität München

Master's Thesis

Efficient approximation of random basis functions for covariance kernels

Author:	Sheng-Chia Yu
Examiner:	Dr. rer. nat. Felix Dietrich
Submission Date:	May 30th, 2023

I hereby declare that this thesis is entirely the result of my own work except where otherwise indicated. I have only used the resources given in the list of references.

May 30th, 2023 Sheng-Chia Yu

Abstract

Kernel methods have been widely acknowledged for their effectiveness in capturing complex nonlinear patterns and structures. However, the computational requirements associated with the kernel matrix have hindered their practical applicability, particularly in large-scale scenarios.

To overcome this limitation, this thesis introduces an approach that utilizes random features to create a feature map. This feature map enables the transformation of training data into a different feature space, facilitating the use of efficient linear learning algorithms while preserving the nonlinear characteristics of traditional kernel methods. By adopting this strategy, the thesis aims to address the computational challenges encountered in the conventional kernel methods.

A detailed implementation strategy for constructing a feature map using random features is presented. The thesis demonstrates the capability of random features by comparing approximated kernel values to exact values. Simulated kernel ridge regression experiments illustrate the reliability of the kernel approximation method, with empirical and theoretical results supporting its effectiveness. However, the limitations of specific feature maps for diverse datasets are also discussed. In addition, a performance comparison between approximated kernels and exact kernels shows the advantage of using the approximation approach. Overall, this thesis offers insights into the use of random features to enhance the practicality and scalability of kernel methods.

Contents

1. Introduction

Kernel methods have long been recognized for their capacity to capture complex nonlinear patterns and structures. However, their practical utility has been constrained by the computational demands associated with the kernel matrix. This thesis addresses this limitation by proposing the use of random features to construct the feature map. By employing this approach, it becomes possible to transform the training data into a different feature space and leverage efficient linear learning algorithms while retaining the nonlinear properties of the original kernel methods. This strategy offers a promising solution to the computational challenges faced by traditional kernel methods, enabling their effective application in large-scale problems.

In chapter 2, this thesis delves into the utilization of kernel methods in machine learning algorithms, focusing on prominent techniques like Kernel Ridge Regression and Support Vector Machines. The chapter provides a comprehensive overview of the kernel method by illustrating its application through a specific example using Kernel Ridge Regression. Additionally, it explores the use of kernels in various other renowned applications within the field of machine learning. By presenting the versatility and effectiveness of kernel methods, this chapter lays the foundation for the subsequent research and experimentation conducted in the thesis.

In chapter 3, a detailed implementation strategy for constructing a feature map for random features such as random Fourier features and random bin features is illustrated. Due to the fact that sometimes an explicit expression of the probability distribution function between a random feature and the approximated kernel does not exist, a numerical scheme for approximating the probability distribution of the parameters of the random features is provided as well. After the critical steps for building a feature map are presented, this thesis demonstrates the ability of the random features by comparing the approximated kernel values to the exact kernel values. Through simulated kernel ridge regression experiments, it is presented that the kernel approximation method is reliable and capable of replacing complex nonlinear learning algorithms with empirical and theoretical results. However, the limitation of a specific feature map for learning various datasets is shown as well. In addition, a performance comparison between random Fourier features approximated Kernel Ridge Regression and exact Kernel Ridge Regression is presented to demonstrate the efficiency of using random features.

Finally, in chapter 4, a comprehensive summary of the results obtained in this thesis is provided. The key findings and contributions of the research are highlighted, emphasizing the advancements made in utilizing random features to address the computational challenges associated with kernel methods. Moreover, the conclusion chapter offers insightful reflections on potential avenues for further improvement and development of the presented techniques.

2. State of the art method for approximation of covariance kernel

In this chapter, we start by providing a comprehensive understanding of kernels and their significance in the kernel method used in machine learning algorithms. The kernel method allows us to capture nonlinear patterns and structures in the data by implicitly mapping the input data to a higher-dimensional feature space. We explore the concept of positive definite kernels and their role in defining the similarity between data points.

In Section 2.2, we delve into the problem of scalability in kernel methods, specifically the challenges associated with computing the kernel matrix or approximating the kernel function. We present several methods to address these challenges, including the Nyström method, which approximates the kernel matrix by sampling a subset of the training data. We also discuss explicit approximation techniques, such as using Taylor series expansions to approximate the kernel function directly.

One prominent approach for kernel approximation is the use of random features. We provide a detailed explanation of this method, which involves generating random features based on the Fourier transform of the kernel function. These random features enable us to approximate the kernel function and transform the input data into a feature space where linear algorithms can be applied. We also introduce an extended method called the Leverage-score-based random Fourier feature, which improves the effectiveness of the random feature selection.

In section 2.3, we showcase the application of kernels in various machine learning algorithms, with a focus on Kernel Ridge Regression and Support Vector Machines. We demonstrate how these algorithms utilize the kernel method to capture complex relationships and make accurate predictions. By using kernels, we can handle diverse data types and exploit nonlinear structures in the data, enhancing the performance of the learning algorithms.

2.1. Kernel

In machine learning algorithms, the prediction accuracy of the model relies heavily on the similarity between the input data points. To quantify this similarity, a kernel or covariance function is utilized to express the statistical relationship between two inputs. Since the kernel function is symmetric and positive definite, it satisfies the requirements of Mercer's theorem. By employing such kernel function, we can effectively and efficiently compute the similarity of inputs, even after transforming them into a higher-dimensional space. This circumvents the need to project the data points to a higher-dimensional space and evaluate dot products, which can be computationally expensive. This technique, known as the kernel method or kernel trick, is widely used in machine learning algorithms. In this section, we will provide an in-depth explanation of the kernel method with a ridge regression problem and introduce several commonly used kernels.

2.1.1. Definition of covariance kernel

Let $\mathcal{X}$ be any space. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called a kernel function if $\forall n \geq 1, x_1, x_2, \ldots, x_n \in \mathcal{X}$ and $c_1, \ldots, c_n \in \mathbb{R}$ we have

$$\sum_{i,j=1}^{n} c_i c_j k(x_i, x_j) \geq 0. \tag{2.1}$$

Given a set of points $x_1, \ldots, x_n \in \mathcal{X}$, we define the corresponding kernel matrix as the Gram matrix $\boldsymbol{K}$ with entries $k_{ij} = k(x_i, x_j)$. The condition above is equivalent to saying that $\boldsymbol{c}'\boldsymbol{K}\boldsymbol{c} \geq 0$ for all $\boldsymbol{c} \in \mathbb{R}^n$ [14].

2.1.2. Kernel method

To demonstrate the kernel method and show its advantage, we illustrate it with a Kernel Ridge Regression problem [21, 5]. In a ridge regression problem, we are provided with a set of data $(x_i, y_i)_{i=1}^n$, where x_i represents the input data points and y_i denotes the corresponding targets. The task is to find a linear function y that models the relationship between x_i and y_i. The model is given by

$$y = w^T x, \tag{2.2}$$

where $x, w \in \mathbb{R}^d$ and $y \in \mathbb{R}$. A simple way to find the function is to minimize the squared error loss,

$$L(w) = \frac{1}{2}\sum_{i=1}^{n}(y_i - w^T x_i)^2 + \frac{1}{2}\lambda \|w\|^2, \tag{2.3}$$

where λ is the regularization parameter to prevent overfitting and is commonly chosen by using the cross-validation algorithm. After taking the derivatives and equating them to zeros gives,

$$\sum_i (y_i - w^T x_i) x_i = \lambda w \tag{2.4}$$

$$\Rightarrow w = (\sum_i x_i x_i^T + \lambda I)^{-1} (\sum_j y_j x_j), \tag{2.5}$$

this can be shown in matrix form as,

$$\Rightarrow w = (X^T X + \lambda I)^{-1} X^T Y, \tag{2.6}$$

and if we replace the x with the feature map $\phi(x) \in \mathbb{R}^m$ and change the linear model to,

$$y = w^T \phi(x), \tag{2.7}$$

the derived weights become,

$$w = (\Phi(x)^T \Phi(x) + \lambda I)^{-1} \Phi(x)^T Y. \tag{2.8}$$

Computing the term $\Phi(x)^T \Phi(x)$ can be computationally expensive, especially when the dimensionality of the feature space is large. To avoid this computational burden, we can derive a new format for the loss function that generates the repetitive term $\Phi(x)\Phi(x)^T$. This term is commonly referred to as the Gram Matrix or Kernel Matrix and is obtained by substituting $\Phi(x)$ for x at the beginning. As a result, the new derivation gives,

$$w = \Phi(x)^T (\Phi(x)\Phi(x)^T + \lambda I)^{-1} Y \tag{2.9}$$

$$= \Phi(x)^T (\boldsymbol{K} + \lambda I)^{-1} Y. \tag{2.10}$$

Since the $\Phi(x)\Phi(x)^T$ is symmetric and positive-definite, based on Mercer's Theorem, it can be expressed as follows,

$$k(x, y) = \langle \phi(x), \phi(y) \rangle. \tag{2.11}$$

Thus, $\boldsymbol{K}$ can be computed using the original data points without transforming them to the feature space or having knowledge of the feature map $\phi(x)$. This is possible because the Gram Matrix is computed solely based on the inner products of the data points, which can be efficiently calculated using the kernel function.

By formula 2.7, we can now make prediction using,

$$y = Y^T ((\Phi\Phi^T + \lambda I)^{-1})^T \Phi\phi(x) \tag{2.12}$$

$$= Y^T ((\Phi\Phi^T + \lambda I)^{-1})^T k_x, \tag{2.13}$$

where $k_x = \Phi\phi(x)$.

2.1.3. Examples of kernels

In addition to the theoretical introduction of kernels and their application in the kernel method, it is important to discuss some common kernels that are frequently used in machine learning algorithms. They have been studied extensively and have been shown to perform well in various applications. Some of the standard kernels include the Gaussian, polynomial, Laplacian, and spline kernels [9].

Polynomial kernels

The polynomial kernel is defined as the dot product of two data points raised to a certain power. A homogeneous polynomial kernel $k(x, x') = \langle x, x' \rangle^p$ is positive definite $\forall p \in \mathbb{N}$ and $x, x' \in \mathbb{R}^d$. An inhomogeneous polynomial kernel is

$$k\left(x, x'\right) = (\langle x, x' \rangle + c)^p, \tag{2.14}$$

where $p \in \mathbb{N}$ and $c \geq 0$.

Gaussian kernels

It is one of the Radial basis functions(RBF). An RBF is a real-value function whose value depends on the distance between the two inputs. One input is usually a fixed point called the center. Define

$$\phi_{\boldsymbol{c}}(\boldsymbol{x}) = \phi(\|\boldsymbol{x} - \boldsymbol{c}\|), \tag{2.15}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is a data point and $\boldsymbol{c} \in \mathbb{R}^n$ is the center.

The Gaussian kernel is a popular choice and is defined as the exponential of the negative squared Euclidean distance between two data points. It is sometimes called "Exponential Quadratic". Since the Gaussian kernel has nice properties such as it can be integrated against most functions, it becomes the default kernel for Gaussian Process and Support Vector Machine [7]. It is given by

$$k(x, x') = \sigma^2 e^{-\gamma \|(x - x')\|_2^2}, \tag{2.16}$$

where $x, x' \in \mathbb{R}^n$ are data points, $\gamma = 1/2l^2$ is lengthscale, and σ^2 is output variance which is a scale factor every kernel has in front.

Laplacian kernels

The difference between Laplacian kernels and Gaussian kernels is that instead of using the square of 2-norm for calculating the distance, it uses 1-norm. It has the form,

$$k(x, x') = \sigma^2 e^{-\gamma \|(x - x')\|_1}. \tag{2.17}$$

2.2. Approximation of covariance kernels

Kernel methods are widely recognized for their ability to learn nonlinear structures, but their practical utility can be limited by scalability issues in large-scale problems due to the high time and space complexities involved. Kernel machines such as Kernel Ridge Regression (KRR) and Support Vector Machines (SVMs) offer the attractive feature of being able to approximate any function or decision boundary arbitrarily well given sufficient training data. However, methods that operate on the kernel matrix (Gram matrix) of the data are known to scale poorly with the size of the training dataset. For example, KRR requires a training time of $O(n^3)$ and space complexity of $O(n^2)$ to store the kernel matrix. These computational demands can render the approach computationally infeasible, particularly when dealing with large n. In addition, training a nonlinear SVM on a dataset with half a million training examples can be a time-consuming process, even on powerful workstations, often taking several days to complete.

To tackle the scalability issue, a series of kernel approximation algorithms have been developed in the past years. One way is designed to approximate the kernel matrix while another way aims to approximate the kernel function directly. Methods designed to approximate the kernel matrix includes greedy basis selection techniques [19], divide-and-conquer approaches [10], and Nyström methods [22]. These methods provide a data-dependent vector representation of the kernel. Another way that approximates the kernel function directly includes random features [16] for kernel approximation and explicit approximation with Taylor series [6]. In particular, the random features method has a widespread impact on kernel approximation.

To reduce the poor scalability of kernel methods, methods approximating the kernel function directly take advantage of the fast linear models by constructing explicit mapping. The linear models can be learned after transforming the training data into a relatively low-dimensional randomized feature space. This reduces the time and memory required while maintaining the power of using the nonlinear kernel method.

In this section methods for kernel approximation are presented. Nyström Method which is one of the popular methods that approximate the kernel matrix is first introduced. It is a data-dependent approach by randomly sampling the training data. Methods that approximate the kernel functions directly such as Explicit approximation with Taylor series are also presented. In subsection 2.2.2, a detailed explanation of random features for kernel approximation is given and followed by a more sophisticated approach called leverage scored-based sampling in subsection 2.2.3.

2.2.1. Kernel approximation methods

Introduction to Nyström Method

The Nyström method is a commonly used technique for approximating kernels with low rank. By subsampling the data on which the kernel is evaluated, an approximation to the eigendecomposition of the Gram matrix can be obtained. This is accomplished by performing an eigendecomposition on a smaller system of size $m < n$, and expanding the results back up to n dimensions [22].

One benefit of the Nyström method is that it only requires computing and storing an $m \times n$ portion of the Gram matrix, rather than the whole matrix. For large-scale problems, good performance can be achieved by using values of only a few hundred. As n grows larger, the ratio m/n can be made even smaller. In addition, kernels with rapidly decaying eigenvalues, such as the Gaussian kernel, are expected to yield particularly accurate approximations.

Introduction to Explicit approximations with Taylor series

Monomial features are obtained by taking a low-order Taylor expansion of the exponential, which results in features that are scaled monomials in the coordinates of the input vectors. The quality of this approximation is evaluated based on its computational cost and is particularly effective for sparse datasets with a moderate number of non-zero dimensions per data point, instead of evaluating the approximation quality by the number of features. This method can be utilized wherever l_2 regularization is used [6].

2.2.2. Random features for kernel approximation

Different from explicit approximations with Taylor series, which selects scaled monomials as features, Rahimi and Recht [16] propose to approximate the target function by randomly selecting finite nonlinear basis functions as features. Here we define the estimator first and then construct the fitting problem with the defined estimator. Consider an estimator approximated by the random features

$$\hat{f} = \sum_{i=1}^{K} c_i \phi(x; w_i), \tag{2.18}$$

where $\phi(x; w) : \mathcal{X} \times \Omega \to \mathbb{R}$ is the nonlinear basis function, $w \in \Omega$ are the parameters of the basis functions, K is the number of the randomly selected features, and $c : \mathcal{C} \to \mathbb{R}$ are the weights of the estimator.

Given a training data set of N pairs of input and target $\{x_i, y_i\}_{i=1\ldots N}$ which are sampled from a target distribution $P(x, y)$ with $x_i \in \mathcal{X}$ and $y_i \in \mathbb{R}$. Consider finding an estimator $f : \mathcal{X} \to \mathbb{R}$ to capture the relationship between inputs and targets given by P, the problem becomes minimize the empirical risk

$$\mathbf{R}[f] \equiv \frac{1}{N} \sum_{i=1}^{N} l(y_i, f(x_i)), \tag{2.19}$$

where l is a loss function that represents the deviation between the estimated results and targets. In Linear Regression, l is the quadratic loss $(y_i - f(x_i))^2$, in Support Vector Machines, l is the hinge loss, $max(0, 1 - y_i f(x_i))$, and in Adaboost, it is the exponential loss $e^{-y_i f(x_i)}$.

Substitute estimator (2.18) into the empirical risk (2.19) we get

$$\operatorname*{argmin}_{w_i \in \Omega, c_i \in \mathcal{C}} \mathbf{R}\left[\sum_{i=1}^{K} c_i \phi(x; w_i)\right]. \tag{2.20}$$

There are two approaches to minimizing the empirical risk, one is first randomly select the parameter w and optimize over c, and another way is jointly optimizing w and c. Here we present the algorithm [18] which first draws w from a distribution $p(w)$ and then optimizes c with fixed w using a convex optimization.

Algorithm 1 The Weighted Sum of Random Kitchen Sinks fitting procedure

Input: Given a data set of N points $\{x_i, y_i\}_{i=1\dots N}$, an integer K, a scalar C, and a bounded feature function $|\phi(x;w)| \leq 1$ with probability distribution $p(w)$ on the parameters of ϕ.
Output: An estimator $\hat{f}(x) = \sum_{j=1}^{K} c_j \phi(x; w_j)$.

1. Draw $w_1, \dots, w_K$ iid from $p(w)$.
2. Mapping the input data points to the feature space: $z_i = [\phi(x_i, w_1), \dots, \phi(x_i, w_K)]^T$.
3. With w fixed, solve the empirical risk minimization problem

$$\underset{c \in \mathbb{R}^K}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} l(y_i, c^T z_i), \tag{2.21}$$

$$s.t. \quad \|c\|_\infty \leq C/K. \tag{2.22}$$

To state that Algorithm 1 has a high probability to return a function, which has a true risk $\mathbf{R}[f]$ near the lowest true risk achievable by an infinite-dimensional class of functions in set $\mathcal{F}_p$, a theorem [18] is provided below. Here the true risk of a function means the expected loss on the unseen test data set generated from the same distribution as the training data set.

Theorem 2.1. *Define the set*

$$\mathcal{F}_p \equiv \left\{ f(x) = \int_\Omega c(w)\phi(x;w)dw \middle| |c(w)| \leq Cp(w)| \right\}, \tag{2.23}$$

where $p(w)$ is a distribution on Ω, and ϕ satisfies $sup_{x,w}|\phi(x;w)| \leq 1$.
Suppose $c(y, f(x)) = c(yf(x))$, with $c(yf(x))$ L-Lipschitz. Then for any $\delta > 0$, if the training data $\{x_i, y_i\}_{i=1\dots N}$ are drawn iid from some distribution P, Algorithm 1 returns a function $\hat{f}$ that satisfies

$$\mathbf{R}\left[\hat{f}\right] - \underset{f \in \mathcal{F}_p}{\operatorname{argmin}} \mathbf{R}[f] \leq O\left(\left(\frac{1}{\sqrt{N}} + \frac{1}{\sqrt{K}}\right) LC\sqrt{log\frac{1}{\delta}}\right) \tag{2.24}$$

with probability at least $1-2\delta$ over the training data set and the choice of the parameters $w_1, \dots, w_K$.

The set $\mathcal{F}_p$ consists of functions whose weights $c(w)$ decay faster than $p(w)$, which can be illustrated with $\phi(x;w)$ as sinusoids features and the Fourier transform of the functions decay faster than $Cp(w)$.

A more theoretical analysis for the guarantees of finding a nonlinear function $\hat{f}$ is conducted by using techniques from probability on Banach Spaces [17]. It provides two theorems of L_2 and L_∞ error bounds respectively for approximating functions in Reproducing Kernel Hilbert Spaces.

The theorems define a new set of mixture of ϕ with finite $\|\cdot\|_p$ norm

$$\mathcal{F}(X, \Theta, \phi, p) \equiv \left\{ f(x) = \int_\Theta c(\theta)\phi(x;\theta)d\theta \,\middle|\, \|f\|_p < \infty \right\}, \tag{2.25}$$

where $X \subset \mathbb{R}^d$, $\Theta = \mathbb{R}^d \times [-\pi, \pi]$, $\phi(x;\theta) = cos(w'x + b)$, $\theta = (w, b)$, p is a fixed probability distribution on Θ, and $\|f\|_p := sup_\theta |\frac{c(\theta)}{p(\theta)}|$.

By assuming throughout that $|\phi(x;\theta)| \leq 1$ for all x and θ, the two theorems show that a given $f \in \mathcal{F}$ can be approximated to resolution $O(\|f\|_p/\sqrt{K})$ by a function

$$\hat{f}(x) = \sum_{i=1}^{K} c_i \phi(x;\theta_i) \; := z_x(\theta)^T c, \tag{2.26}$$

where $\theta_1, \ldots, \theta_K$ are sampled iid from $p(\theta)$. It is different from equation 2.18 with a shift parameter b.

Up until this point, we have covered the fitting process for obtaining the estimator as well as the theoretical guarantees associated with it. In order to construct the randomized feature map z_i outlined in Algorithm 1, three distinct feature functions, random Fourier, random bin, and random stump denoted as ϕ are introduced [16, 17]. These feature functions play a crucial role in the construction of the randomized feature map and contribute to the overall performance of the linear learning algorithms.

Among the three features discussed, random Fourier features have gained significant popularity due to their ability to find a suitable probability distribution by leveraging the Fourier transform of the approximated kernel. This concept has served as a foundation for numerous research endeavors in kernel approximation. The survey paper [12] primarily centers on random Fourier features and classifies kernel approximation algorithms into two main categories: data-independent algorithms and data-dependent algorithms. The classification is based on whether the selection of feature parameters θ is independent of the training data.

On the one hand, in the data-independent algorithm category, different sampling strategies are employed, including Monte Carlo sampling, Quasi-Monte Carlo sampling, and Quadrature-based methods. On the other hand, data-dependent algorithms utilize information from the training data to guide the selection of feature parameters, aiming to enhance the quality of approximation and potentially improve the generalization performance of learning algorithms based on these features. This category can be further subdivided into three classes: Leverage score sampling, Re-weighted random feature selection, and Kernel learning by random features. In this thesis, particular emphasis is placed on the leverage score sampling method, and a comprehensive theoretical background of this approach will be presented in the subsequent subsection.

2.2.3. Leverage score-based sampling of random features

In addition to sampling parameters from the plain RFF sampling scheme with the probability distribution function $p(\theta)$, another approach is to sample the parameters from an importance-weighted probability distribution function $q(\theta)$. This is known as the weighted RFF sampling scheme.

$$\hat{f}_q(x) = \sum_{i=1}^{K} \alpha_i \phi_q(x;\theta_i) \ := z_{q,x}(\theta)^T \alpha, \tag{2.27}$$

which is similar to Equation 2.26 , except that the feature vectors are sampled from $q(\theta)$ instead of $p(\theta)$. Specifically, $\phi_q(x;\theta_i) = \sqrt{p(\theta_i)/q(\theta_i)}\phi(x;\theta_i)$ and $z_{q,x}(\theta) = [\phi_q(x;\theta_1),\dots,\phi_q(x;\theta_K)]^T$. Therefore, a kernel matrix $\boldsymbol{K} \in \mathbb{R}^{n\times n}$ can be approximated by

$$\boldsymbol{K}_q = \mathbf{Z}_q \mathbf{Z}_q^T, \tag{2.28}$$

where $\mathbf{Z}_q \in \mathbb{R}^{n\times K}$ with $z_{q,x}(\theta)^T$ as its rows.

In contrast to the plain RFF sampling scheme, which primarily aims to approximate the leading eigenvalues of the kernel matrix, sampling from a weighted distribution like $q(\theta)$ is designed to generate Fourier features that cover the entire eigenspectrum of $\boldsymbol{K}$.

The objective is to ensure that the weighted RFFs capture a broader range of eigenvalues, rather than focusing solely on the dominant ones. This can be beneficial in applications where a more comprehensive representation of the kernel matrix is desired, allowing for a richer approximation of the kernel function. By sampling from a weighted distribution that spans the eigenspectrum, the weighted RFFs provide a more balanced and diverse set of features, leading to improved approximation capabilities across the entire range of eigenvalues.

In order to create an importance-weighted probability distribution $q(\theta)$ for random Fourier features, the concept of leverage scores, originally introduced for landmark selection in the Nyström method is utilized. The leverage score function of the integral operator given by the kernel function and the data distribution, as proposed by Bach and Jordan [2], provides a measure of the contribution of each data point to the kernel matrix. Building upon this notion, Avron and Toledo [1] developed the ridge leverage score function based on relating the sampling density to an appropriately defined ridge leverage function with respect to a fixed input data set. The definition of the ridge leverage function is given by

$$l_\lambda(\theta) = p(\theta) z_\theta(\mathbf{x})^T \left(\boldsymbol{K} + n\lambda\mathbf{I}\right)^{-1} z_\theta(\mathbf{x}), \tag{2.29}$$

where $z_{\theta_j}(\mathbf{x})$ is the j-th column of $\boldsymbol{Z} \in \mathbb{R}^{n\times s}$ with $z_{p,x}(\theta)^T$ as its rows and λ is the regularization parameter.

A notable characteristic of the function $l_\lambda(\theta)$ is its correlation with the effective number of parameters,

$$d_{\boldsymbol{K}}^{\lambda} := \int_\Theta l_\lambda(\theta) d\theta = Tr\left[\boldsymbol{K}\left(\boldsymbol{K} + n\lambda\mathbf{I}\right)^{-1}\right], \tag{2.30}$$

where $d_{\boldsymbol{K}}^{\lambda}$ is recognized as a factor in determining the effective number of independent parameters in a learning problem.

With the above notations, the weighted probability distribution $q(\theta)$ in [1] is given by

$$q(\theta) = l_{\lambda}(\theta)/d_{\boldsymbol{K}}^{\lambda}. \tag{2.31}$$

The leverage-scored sampling scheme has primarily focused on providing a faster algorithm to approximate the kernel matrix, as Equation 2.29 involves the inversion of an $n \times n$ kernel matrix. This scheme offers the advantage of requiring fewer Fourier features while still providing theoretical guarantees. It has been shown to converge for learning risk rates in various algorithms, including kernel ridge regression, kernel support vector machines, and kernel logistic regression. In the study conducted by [11], a fast algorithm is proposed that only requires the inversion of a smaller $s \times s$ kernel matrix, enabling efficient generation of samples from the approximated leverage distribution.

A later approach, known as Surrogate Leverage Score-RFF, introduces a surrogate leverage function that avoids the need for matrix inversion altogether [13]. This technique further enhances the efficiency of leverage-scored sampling in approximating the kernel matrix.

2.3. Applications in machine learning

Kernel methods are widely utilized in various machine learning algorithms to capture the nonlinear characteristics of data in higher-dimensional spaces. While several algorithms make use of the kernel method, this section specifically highlights Kernel Ridge Regression (KRR) and Support Vector Machines (SVM) due to their prominence in leveraging kernel methods. However, it is important to note that other popular algorithms, such as Kernel Principal Component Analysis (PCA), Kernel Canonical Correlation Analysis (CCA), Kernel K-means, and Gaussian Process, also heavily rely on the kernel method for their effective operation. These algorithms demonstrate the versatility and effectiveness of kernel methods in various domains and provide powerful tools for capturing complex patterns and structures in data.

2.3.1. Kernel ridge regression

In this section, we give an overview of Kernel Ridge Regression since in section 2.1.2 we have demonstrated the kernel methods by using it as an example.

KRR is a popular regression algorithm that extends linear ridge regression to nonlinear problems using kernel methods. It combines the principles of ridge regression with the ability of kernel methods to capture complex relationships in the data. In traditional ridge regression, the goal is to find a linear function that minimizes the sum of squared errors between the predicted values and the actual values. However, in KRR, nonlinear mapping is applied to transform the original data into a high-dimensional feature space. This transformation allows for the use of a wider range of functions to model the data.

The key idea behind KRR is to introduce a kernel function that computes the similarity between pairs of data points in the feature space. This kernel function implicitly maps the data into a higher-dimensional space, where linear regression can be performed efficiently. By incorporating the kernel function into the regression framework, KRR is able to capture nonlinear relationships between the input variables and the target variable.

To prevent overfitting and ensure robustness, KRR introduces a regularization term that controls the complexity of the model. This regularization term, similar to ridge regression, helps to balance the trade-off between fitting the training data well and maintaining generalization performance on unseen data.

KRR offers several advantages, including its flexibility in capturing nonlinear patterns, its ability to handle high-dimensional feature spaces, and its effectiveness in dealing with small and noisy datasets. It has been successfully applied in various domains, including finance, bioinformatics, and image processing, where complex relationships and nonlinearity are common.

2.3.2. Support vector machines

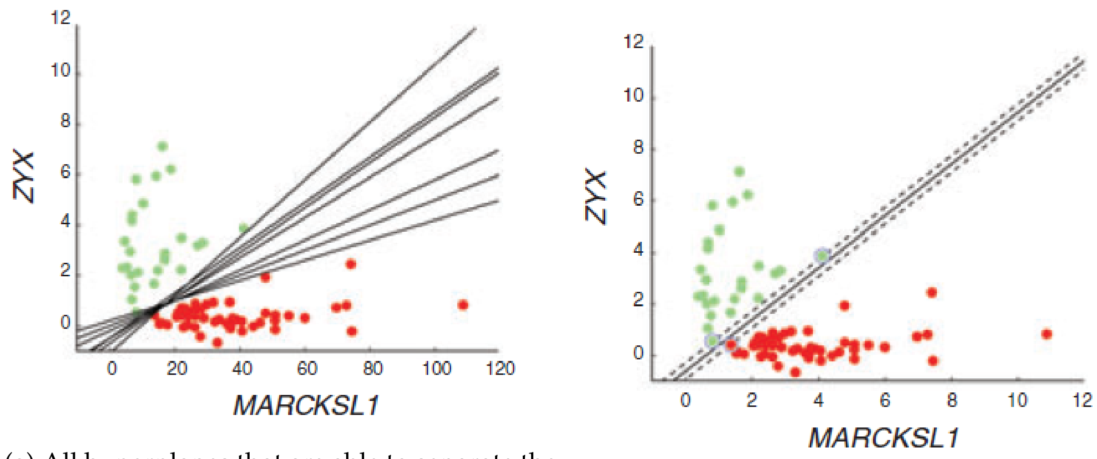
Support Vector Machines (SVM) are widely recognized as one of the most prominent applications of kernel methods. By leveraging the kernel method, SVM extends traditional linear classification to non-linear classification tasks. It is highly effective for both classification and regression problems. In Figure 2.1, we visualize a linearly separable dataset in a 2D space, there are numerous lines that can accurately separate the data. However, SVM selects the line with the maximum distance to the nearest data point, aiming to find the optimal decision boundary. The intuition behind this approach is that by maximizing the margin, which stands for the distance between the decision boundary and the nearest data point, SVM enhances its ability to make accurate predictions. In higher-dimensional spaces, the decision boundaries in SVM are known as hyperplanes. These hyperplanes are defined by a set of points, called support vectors, which are the data points closest to the decision boundaries. The support vectors play a crucial role in SVM because they determine the location and orientation of the hyperplanes.

Consider a training set with two classes composed of N input-output pairs $(x_i, y_i)_{i=1}^{N}$, where x_i represents the input data points and $y_i \in \{-1, 1\}$ denotes the corresponding labels [3]. The task is to find a hyperplane y that is able to separate x_i into the correct label. The decision function is given by

$$y = sgn\left(\langle w, x\rangle + b\right), \tag{2.32}$$

where w and b is the parameter of the hyperplane, and by calculating the value, $(\langle w, x\rangle + b)$, we can decide the label of the input x based on whether the value is negative or positive. We define the margin on either side of the hyperplane to satisfy $\langle w, x\rangle + b = \pm 1$, and we can derive the margin between two classes is at least $\frac{2}{\|w\|}$.

To have the correct condition of classification by having a hyperplane with the largest margin, the following constraints need to be added to make the data points on the correct



(a) All hyperplanes that are able to separate the data.

(b) The hyperplane with the largest margin.

Figure 2.1.: This figure was generated by the gene data of ZYX and MARCKSL1 for illustrating the concept of hyperplanes [15] .

side and outside of the margin. Thus, searching for the optimal hyperplane turns out to be an optimization problem of the following form:

$$\underset{w,b}{\operatorname{argmin}} \frac{1}{2}\|w\|^2, \tag{2.33}$$

subject to

$$y_i \left(\langle w, x_i \rangle + b \right) \geq 1, \tag{2.34}$$

which is referred to as the primal problem. To solve an optimization problem with inequality constraints, the Lagrange method is adopted, and the new problem formulation is given by

$$\underset{\alpha}{\operatorname{argmax}} \left(\underset{w,b}{\operatorname{argmin}} L(w,b,\alpha) \right), \tag{2.35}$$

where

$$L(w,b,\alpha) \equiv \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left(y_i \left(\langle w, x_i \rangle + b \right) - 1 \right), \tag{2.36}$$

with $\alpha_i \geq 0$. In the formulation, we have N Lagrange multipliers for each inequality constraint. In the end, solving the dual optimization problem by plugging the result of the partial derivative of Equation 2.36 with respect to w and b back to itself, we can have the decision function as

$$f(x) = sgn \left(\sum_{i=1}^{N} \alpha_i y_i \langle x, x_i \rangle + b \right). \tag{2.37}$$

The value of α_i can be interpreted as that data points with $\alpha_i > 0$ are the support vectors and data points with $\alpha_i = 0$ are ignored by the hyperplane. We can further replace the dot product with a kernel to find a nonlinear hyperplane or use the feature map discussed in this thesis to transform the data points to other features space and then apply this linear SVM. The decision function with a covariance kernel is given by

$$f(x) = sgn\left(\sum_{i=1}^{N} \alpha_i y_i k(x, x_i) + b\right). \tag{2.38}$$

3. Efficient approximation of random basis functions for covariance kernels

In this chapter, we begin by establishing the quantitative metrics that will be utilized in our experiments. To provide clarity regarding the implementation of these experiments, we present a detailed description of the crucial steps involved in constructing specific feature maps for each random feature in section 3.2. In order to showcase the ability of random features to approximate the corresponding feature maps, in section 3.3, we compare the kernel values computed using the constructed feature maps with the exact kernel values. Moreover, we employ the feature maps in linear ridge regression to validate the convergence of the excess risk. Simultaneously, depending on the required accuracy of each learning task, we demonstrate that the kernel approximation method is reliable and capable of replacing complex nonlinear learning algorithms with empirical and theoretical results.

3.1. Definition of performance and efficiency measurements for feature mapping

3.1.1. Root-Mean-Square-Error

In section 3.3, the primary objective of this thesis is to generate a precise approximation of a kernel value using a feature map $\phi(x)$. However, the effectiveness of these algorithms can only be evaluated if we can assess the error between the exact kernel and its approximation. In general, various types of errors are employed to measure this discrepancy. In this chapter, we specifically consider the following error metric:

$$RMSE = \sqrt{\frac{\sum \left(k(x,y) - \hat{k}(x,y)\right)^2}{N}}. \tag{3.1}$$

The exact kernel value is represented as $k(x, y)$, while its approximation is denoted as $\hat{k}(x, y) = \langle \phi(x), \phi(y) \rangle$.

The absolute error measure alone may be challenging to interpret without considering the scale of the data. To address this, the evaluations described in section 3.3 are using a normalized form of error as well. This normalized error is obtained by dividing the error from equation 3.1 by the mean of the exact kernel values $\bar{k}$. By employing the normalized error, we can better understand the error in relation to the scale of data. The complete expression for the normalized error is provided below:

$$Normalized \quad RMSE = \frac{RMSE}{\bar{k}}. \tag{3.2}$$

These error metrics allow us to quantify the dissimilarity between the exact kernel value and its approximation, enabling us to assess the accuracy and fidelity of the proposed methods.

3.1.2. Coefficient of determination

In section 3.3.3, the Coefficient of determination R^2 is used as a metric to evaluate how well the model is trained with the data. The R^2 score provides a measure of the proportion of the variance in the dependent variable that is predictable from the independent variables. It indicates the goodness of fit of the model and represents the percentage of the response variable's variance that is explained by the predictor variables. By calculating the R^2 score, we can gauge how well the model predicts the observed data. It provides valuable information about the model's ability to capture the underlying relationships and patterns in the data. The R^2 score ranges from 0 to 1, where a score of 0 indicates that the model cannot predict the response variable, a score between 0 and 1 suggests partial predictability and a score of 1 signifies a perfect prediction. It is given by

$$R^2 = 1 - \frac{RSS}{TSS} \tag{3.3}$$

$$= 1 - \frac{\sum_i (y_i - f_i)^2}{\sum_i (y_i - \bar{y})^2}, \tag{3.4}$$

where RSS is the residual sum of squares, TSS is the total sum of squares, y_i is the ground truth of observed data, f_i is prediction, and $\bar{y}$ denotes the mean of the ground truth values.

Compared to other metrics like Mean Squared Error (MSE) and Root Mean Squared Error (RMSE), the R^2 score offers additional insights by expressing the prediction performance as a percentage. This makes it more interpretable and facilitates comparisons between different models.

3.2. Approximation of covariance kernels with random features

In this section, we delve into the implementation details of the approximation process. Specifically, we showcase the methodology for approximating various kernels through the utilization of random Fourier features and bin features. To accomplish this, we define the features and their corresponding parameters. Subsequently, following Algorithm 1, it becomes necessary to establish an appropriate probability distribution, denoted as $p(\theta)$, to facilitate the random sampling of a specific number of features for constructing the feature map z.

The determination of the probability distribution can vary depending on the specific features employed. One approach involves obtaining an explicit solution for $p(\theta)$ by solving the predefined relationship between the kernel and the features. This allows for direct characterization of the probability distribution, ensuring its suitability for the random feature selection process. Thus, the kernel can be approximated by

$$k(\delta) = \int_{\Theta} p(\theta)\phi(\delta;\theta)d\theta, \tag{3.5}$$

where $\delta = x - y$ and $\theta \in \Theta$ is the parameter of the nonlinear feature function ϕ.

The second approach involves calculating the convolution of the target kernel and the selected features. This can be achieved by either finding an explicit solution or resorting to numerical integration methods if an explicit solution is not available. As an example, this section introduces Gaussian quadrature, a numerical integration technique, and provides its implementation details.

Upon obtaining the feature map, the next step involves transforming the original data points into the feature space. This transformation enables various applications, such as approximating the desired kernel or utilizing the transformed features for linear learning algorithms.

3.2.1. Probability distribution of random feature parameters

The probability distribution $p(\theta)$ for random Fourier features and bin features has been explicitly derived in previous research. However, for features without a theoretical derivation, we can approximate the distribution by convolving the desired kernel with the features.

When it comes to generating parameters from a normal or uniform distribution, we can readily utilize the available scientific packages in Python. However, generating parameters from an arbitrary distribution poses more challenges. Initially, we need to construct a histogram of the distribution, which is straightforward for distributions with explicit expressions. In cases where an explicit form is unavailable, the histogram must be generated using numerical methods. It's important to note that the quality of the numerical approximation can significantly impact the resulting histogram and subsequent usage.

Finally, the generated histogram can serve as input for creating a continuous distribution using a SciPy module specifically designed for creating univariate distributions [20]. This allows us to work with the desired distribution for further computations and analyses.

Random Fourier features

This feature first projects the data point to a randomly chosen line drawn from $p(w)$ and then passes the result scalar from the inner product to a sinusoid after plus an offset to it.

With Bochner's theorem, if the approximated kernel is properly scaled, its Fourier transform is a proper probability distribution. Then, we can use its Fourier transform as the distribution $p(w)$ and define [16]

$$k(\mathbf{x}-\mathbf{y}) = \int_{\mathcal{R}^d} p(w)e^{jw'(\mathbf{x}-\mathbf{y})}dw = E_w\left[\zeta_w(\mathbf{x})\zeta_w(\mathbf{y})^*\right], \tag{3.6}$$

where $\zeta_w(\mathbf{x})\zeta_w(\mathbf{y})^*$ is an unbiased estimate of $k(\mathbf{x}-\mathbf{y})$ when w is drawn from p.

Since both the kernel value $k(\Delta)$ and the probability distribution $p(w)$ are real values, $e^{jw'\Delta}$ can be replaced with $cos(w'\Delta)$. To give a real-valued mapping, the unbiased estimate can be defined as either $\phi_w(\mathbf{x}) = [cos(w'\mathbf{x}) \quad sin(w'\mathbf{x})]'$ since $\phi_w(\mathbf{x})'\phi_w(\mathbf{y}) = cos(w'\Delta)$ or $\phi_w(\mathbf{x}) = \sqrt{2}cos(w'\mathbf{x}+b)$, where b is drawn uniformly from $[0, 2\pi]$.

Random bin features

The underlying concept of the algorithm involves repeatedly dividing the input space using a randomly shifted grid, with a resolution that is randomly selected. Each point in the input space, denoted as x, is then encoded as a binary indicator $\phi(x)$ over the bins [16].

In the context of a one-dimensional kernel, the approach involves partitioning the real number line using a grid with a randomly drawn pitch δ as the grid resolution. The grid is also randomly shifted by an amount u within the range of $[0, \delta]$. As a result, the real line is divided into intervals such as $[u+n\delta, u+(n+1)\,\delta]$ for all integers n.

For each data point, we can determine the bin it falls into by calculating the bin number using $\hat{x} = \lceil \frac{x-u}{\delta} \rceil$. This leads to the definition of a hat kernel $\hat{k}(x, y; \delta) = max\left(0, 1 - \frac{|x-y|}{\delta}\right)$, which represents the likelihood of two data points falling within the same bin. It is defined as follows:

$$Pr_u = [\hat{x} = \hat{y}|\delta] = \hat{k}\,(x, y; \delta)\,. \tag{3.7}$$

The hat kernel $\hat{k}$ indicates the probability that x and y fall in the same bin, given a particular value of δ.

By repeatedly dividing the input space using the described grid construction, we can approximate the shift-invariant kernel, denoted as $k(x, y)$. This approximation is obtained by integrating the hat kernel over a range of pitch values δ, weighted by the probability density function $p(\delta)$:

$$k\,(x, y) = \int_0^\infty \hat{k}\,(x, y; \delta)\, p\,(\delta)\, d\delta. \tag{3.8}$$

In this equation, δ is sampled from the probability density function $p(\delta)$, and u is sampled from the range $[0, \delta]$. The resulting probability of x and y falling into the same bin corresponds to the kernel value $k(x, y)$. The choice of $p(\delta)$ can be determined by setting $p(\delta) = \delta\ddot{k}(\delta)$, based on Lemma 1 in the proof [16].

For separable multivariate data points $\mathbf{x} \in \mathbb{R}^d$, the shift-invariant kernel can be expressed as the product of individual kernel functions $k(x, y)$ in each dimension. In other words, if $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ and $\mathbf{y} = (y_1, y_2, \ldots, y_d)$, then the shift-invariant kernel $k(\mathbf{x}, \mathbf{y})$ is given by

$$k(\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{d} k(x_i, y_i). \tag{3.9}$$

This formulation allows for the independence of each dimension when evaluating the kernel function.

3.2.2. Implementation of feature mapping

The construction of the feature map z in Algorithm 1 can be adapted to various formats depending on the chosen nonlinear function and its variations. In this thesis, we present step-by-step algorithms for each random feature to illustrate the implementation of creating a feature map.

By following these algorithms, researchers and practitioners can easily replicate the process and generate the desired feature maps. The step-by-step instructions provide clarity and facilitate the implementation of different nonlinear functions and their variations, enabling further experimentation and exploration in the field.

Random Fourier features

To decrease the variance, we can sample multiple times to obtain more feature functions and normalize the resulting vector. Specifically, we can concatenate D randomly selected ϕ_w functions into a column vector z and then divide each component by $\sqrt{D}$. Algorithm 2 describes each step to generate a feature map with $\phi_w(\mathbf{x}) = \sqrt{2}cos(w'\mathbf{x} + b)$ for data point transformation [16, 4].

Algorithm 2 Random Fourier feature map

Input: A positive-definite shift-invariant kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$.
Output: A feature map with randomly chosen parameters $z(\mathbf{x}) : \mathcal{R}^d \to \mathcal{R}^D$.

1. Compute the cosine transform $p(w)$ of the kernel

$$p(w) = \frac{1}{2\pi} \int cos(w'\Delta + b)k(\Delta)d\Delta.$$

2. Draw D iid samples $w_1, \ldots, w_D \in \mathcal{R}^d$ from the probability density function p in cosine space and b uniformly from $[0, 2\pi]$.

3. Let $z(\mathbf{x}) \equiv \sqrt{\frac{2}{D}}\,[cos(w_1'\mathbf{x} + b_1), \ldots, cos(w_D'\mathbf{x} + b_D)]'$.

After creating the feature map according to Algorithm 2, the kernel approximation is given by the sample average of $\phi_{w_j}(\mathbf{x})\phi_{w_j}(\mathbf{y})$ over all w_j in the feature space. As a result, the approximation is given by

$$z(\mathbf{x})'z(\mathbf{y}) = \frac{1}{D}\sum_{j=1}^{D} \phi_{w_j}(\mathbf{x})\phi_{w_j}(\mathbf{y}). \tag{3.10}$$

Random bin features

Using the same method as in the implementation of a random Fourier feature map, we can reduce the variance of the estimated result by concatenating D binary indicator vectors $\phi(\mathbf{x})$ into a list of binary indicators z and scaling them by $\sqrt{\frac{1}{D}}$. This normalization helps to ensure that the resulting feature map has a unit norm and reduces the impact of variations in the number of features.

Algorithm 3 Random bin feature map

Input: A positive-definite shift-invariant kernel $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{x} - \mathbf{y})$ given in equation 3.9 with data points $\mathbf{x} \in \mathbb{R}^d$ and $p(\delta) \equiv \delta\ddot{k}(\delta)$ is a proper probability distribution on $\delta > 0$.
Output: A feature map with randomly chosen parameters $z(\mathbf{x}) : \mathcal{R}^d \to \mathcal{R}^{D\times d}$.

```
1. Draw grid parameters $\Delta, \mathbf{u} \in \mathbb{R}^d$ with the pitch $\delta_i$ from $p(\delta)$ and $u_i$ from the uniform distribution within the range $[0, \delta_i]$.
2. Let the feature be the coordinate of the bin containing $\mathbf{x}$ as a vector $\phi(\mathbf{x}) \equiv \left(\left\lceil \frac{x_1-u_1}{\delta_1} \right\rceil, \dots, \left\lceil \frac{x_d-u_d}{\delta_d} \right\rceil\right)'$.
3. Repeat the process for $D$ time and concatenate $D$ feature $\phi(\mathbf{x})$ into $z(\mathbf{x}) \equiv \sqrt{\frac{1}{D}}\,[\phi_1(\mathbf{x}), \dots, \phi_D(\mathbf{x})]'$.
```

To utilize the feature map defined in Algorithm 3, instead of creating a binary indicator vector, we can calculate the approximated kernel value by comparing the values in two feature maps element-wise to determine if the corresponding data points are in the same bin. This comparison results in a binary map indicating bin equality. We then sum up the elements in each dimension of the map separately, scale the sums by $\frac{1}{D}$ to obtain the individual approximations $k(x_i, y_i)$, and finally calculate the product of all dimensions to obtain the overall approximation result. This approach allows us to compute the approximation of the kernel value based on the binning of data points and their corresponding feature maps.

3.3. Evaluation of approximation result

The quality assessment of the feature map has been a focus of several studies, with an emphasis on determining the number of random features necessary to achieve a satisfactory approximation of the target kernel value. This evaluation metric serves as a quantitative measure of the feature map's ability to capture the essential characteristics of the kernel.

Another evaluative approach involves assessing the number of random features required to achieve a comparable expected risk to that of a learned estimator while maintaining accuracy and precision. This evaluation criterion provides valuable perspectives into the efficiency and performance of the feature map relative to other learning algorithms [12].

Furthermore, these evaluations shed light on the trade-off between computational complexity and approximation accuracy in kernel approximation methods. They also provide practical guidance regarding the number of random features needed to achieve desired levels of performance in various application scenarios.

In addition to the aforementioned evaluation metrics, it is also common to analyze the behavior of the approximation error as the number of data points increases. This analysis helps to understand how the approximation performance scales with the dataset size and provides insights into the limitations and generalizability of the kernel approximation methods.

By considering these evaluation approaches and metrics, we can gain a comprehensive understanding of the capabilities and limitations of random feature methods in capturing the properties of kernels. This knowledge is valuable for selecting appropriate methods and determining the necessary computational resources for specific applications.

3.3.1. Comparison with exact kernels

In this section, our experiments focused on assessing the ability of random features to capture the characteristics of kernels. To accomplish this, we calculated the probability distribution of feature parameters using either an explicit form or Gaussian quadrature as an approximation method. The primary goal was to approximate the kernel values using the induced feature map and compare the effectiveness of different features in approximating the same kernel. In the case of random Fourier features, two approaches were employed: the plain method and the leverage score method, as discussed in subsection 2.2.3 with modified source code [11].

We can gain an understanding of the strengths and weaknesses of different approaches in capturing the essential properties of kernels by comparing the results obtained from various feature representations. This analysis enables us to understand which feature maps are better suited for specific types of kernels and provides valuable information for selecting appropriate feature representations in practical applications.

For these experiments, 200 one-dimensional samples were drawn uniformly from the interval $[0, 1]$. The evaluation of kernel values was performed by computing them for each pair of samples in the dataset. We then compared the obtained kernel values with the desired values and measure the accuracy of the approximation using root-mean-square error (RMSE) and normalized RMSE (NRMSE) metrics. These metrics provide insights into the quality of the approximation as the number of feature components increases.

To compare the plain RFF method and the leverage score RFF method, a pool of weights was created at the beginning. In the plain method, the features were selected based on the original order of the weights. However, in the leverage score method, the weights were sorted according to the weighted probability distribution $q(\theta)$ mentioned in Equation 2.31. The features were then selected based on this new order of weights. By using different selection strategies, the leverage score method aims to improve the effectiveness and efficiency of the feature selection process in approximating the kernel matrix.

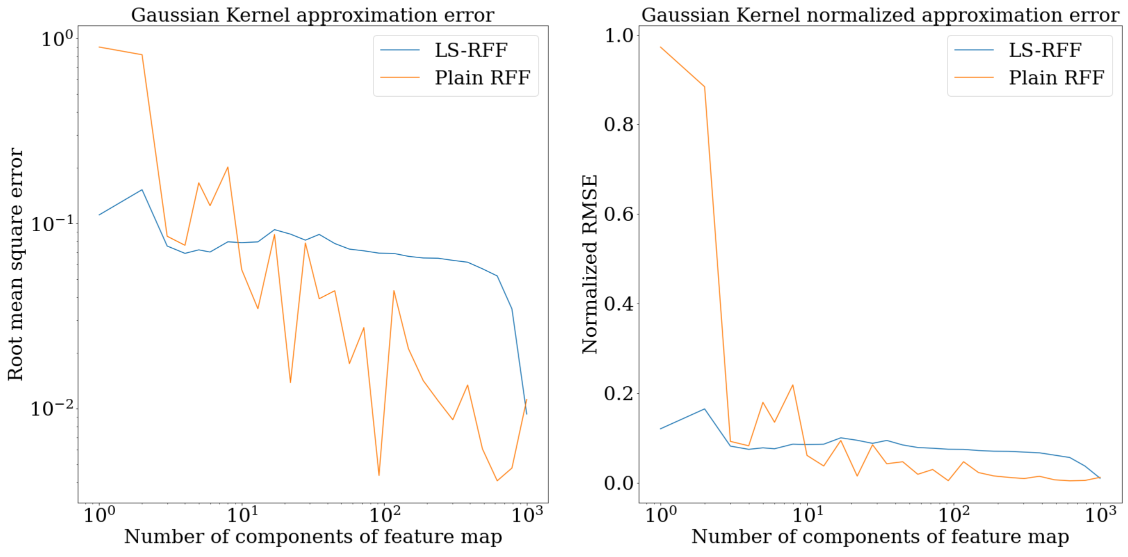
Through tracking the RMSE and NRMSE values across different numbers of feature components, we can analyze how the accuracy of the approximation improves with the inclusion of more features. This information allows us to understand the trade-off between computational complexity and the accuracy of the kernel approximation.

We present the results of the evaluation, including plots and quantitative analysis of the RMSE and NRMSE values, as the number of feature components varies. This analysis provides a comprehensive understanding of how the approximation performance evolves and whether increasing the number of feature components leads to significant improvements. By conducting these experiments, we can assess the effectiveness of random features in approximating kernel values and acquire observation into their ability to capture the essential properties of the kernels. The results of these experiments will be presented and analyzed in subsequent sections.

Gaussian kernel

The Fourier transform of a Gaussian kernel, defined as $k(x, y) = e^{-\gamma |x-y|_2^2}$, can be obtained by taking the Fourier transform of the Gaussian function. The Fourier transform $p(w)$ of a Gaussian kernel is known to be a Gaussian distribution with mean 0 and covariance $2\gamma I$, where I represents the identity matrix. This distribution can be simply emulated by using API in the Numpy library [8]. The feature map, in this case, was generated using the RBFSampler from the scikit-learn library [4] with $\gamma = 0.5$. RBFSampler is a feature map that approximates the Radial Basis Function (RBF) kernel. It is commonly used for dimensionality reduction and non-linear feature transformation in machine learning tasks. RBFSampler works by randomly generating a set of weights and offsets for each dimension of the input data. It then applies a nonlinear transformation to the input data using the generated feature map.

In Figure 3.1, the results demonstrate that as the number of components increases, the approximation accuracy improves. This indicates that using more random features in the feature map leads to a better approximation of the target kernel. Both the plain method and the leverage score method show that random Fourier features can effectively approximate the Gaussian kernel. The key factor for achieving a good approximation is ensuring a



(a) RMSE

(b) Normalized RMSE

Figure 3.1.: Approximation error of Gaussian Kernel value with random Fourier features.

sufficient number of random features.

Furthermore, the leverage score method, which is a specific approach for selecting random features, exhibits a superior ability to capture the essential characteristics of the kernel. This method achieves comparable accuracy with fewer components compared to the plain method. By utilizing the leverage scores, the feature selection process is optimized to choose the most informative features, resulting in a more efficient and effective approximation of the kernel. However, as the number of components increases, the plain RFF method outperforms the leverage score RFF method. Eventually, both methods utilize all the weights in the pool and achieve the same approximation accuracy. An interesting observation is that since the leverage score method changes the order of the weights w, the pairing with offsets b is also different, resulting in different results despite using the same weights in the end.

Laplacian kernel

Both random Fourier features and bin features are capable of finding a suitable probability distribution for the Laplacian kernel, which ensures that the resulting function is positive everywhere and can be used to construct a random feature map. In the case of random Fourier features, the Fourier transform can be directly computed to determine the appropriate probability distribution. On the other hand, for bin features, Algorithm 3 can be used to calculate the probability distribution based on the derived formula $p(\delta) = \delta\ddot{k}$. This formula enables the generation of bin features that accurately capture the characteristics

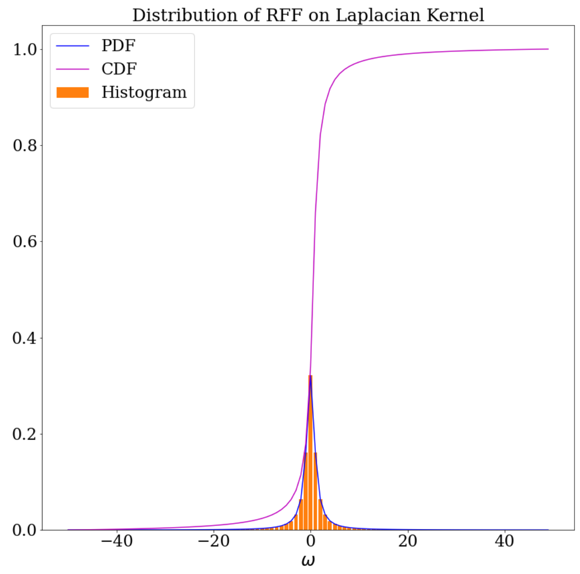


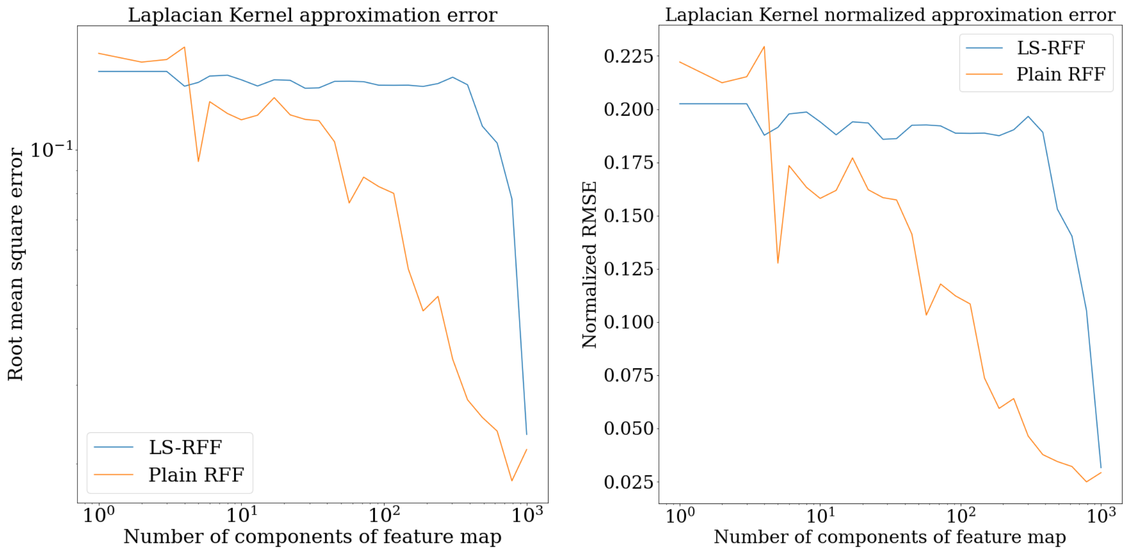
Figure 3.2.: The probability distribution $p(w)$ of the Laplacian kernel is generated by creating a histogram of its Fourier transform values. The Fourier transform is sampled evenly on the interval $[-50, 50]$ with 101 sampled values.

of the Laplacian kernel.

In Figure 3.2, the Fourier transform of the Laplacian kernel, which is given by $\frac{1}{\pi(1+\omega^2)}$, is utilized to generate a probability distribution. This is achieved by using the sampled values as a histogram with the help of the SciPy random variate histogram module [20]. By using a sufficient number of sampled values and a suitable range, the histogram can provide an approximation of the true probability distribution. The histogram captures the frequency of occurrence of different values in the Fourier transform, giving insights into the shape and characteristics of the probability distribution. The resulting distribution represents the desired probability distribution for generating random Fourier features that approximate the Laplacian kernel.

To verify the generated distribution, it can be compared to the probability density function (PDF) and cumulative distribution function (CDF) of the generated distribution function. The PDF represents the probability density at each point, while the CDF provides the probability of observing a value less than or equal to a given threshold. By comparing the generated distribution with the PDF and CDF, we can assess the accuracy of the generated distribution and ensure that it closely matches the desired distribution. This comparison serves as a validation step to confirm that the generated distribution aligns with the expected properties of the Laplacian kernel.

Figure 3.3 illustrates that while both methods achieve accurate approximations of the



(a) RMSE (b) Normalized RMSE

Figure 3.3.: Approximation error of Laplacian Kernel value with random Fourier features.

Laplacian kernel in the end, the leverage score RFF method does not initially outperform the plain RFF method in terms of capturing the characteristics of the kernel. However, an interesting observation is that there is a distinct transition point at around using 500 components in the experiment where the LS-RFF method starts to demonstrate better performance compared to the previous approximation.

Figure 3.4 displays the probability distribution of the pitch δ for bin features employed in the approximation of the Laplacian kernel. The distribution initiates from zero due to the restriction that the pitch δ must be positive and exhibits a peak in density near zero. Subsequently, it experiences a sharp decline and diminishes towards zero as it is based on the exponential function $\delta e^{-\delta}$.

Figure 3.5 illustrates the convergence behavior of the approximation error when using bin features to approximate the Laplacian kernel. The normalized root mean square error (NRMSE) of the bin feature decreases rapidly and reaches 10 percent with approximately 10 components. In contrast, in Figure 3.3, the Fourier feature method requires approximately 10 times the number of components to achieve the same level of accuracy. However, it is important to note that the bin feature approximation reaches a saturation point and cannot further improve the accuracy by increasing the number of features.

This observation suggests that when the desired accuracy level is relatively lower, using bin features can offer computational resource savings compared to Fourier features while still providing reasonable approximation quality.

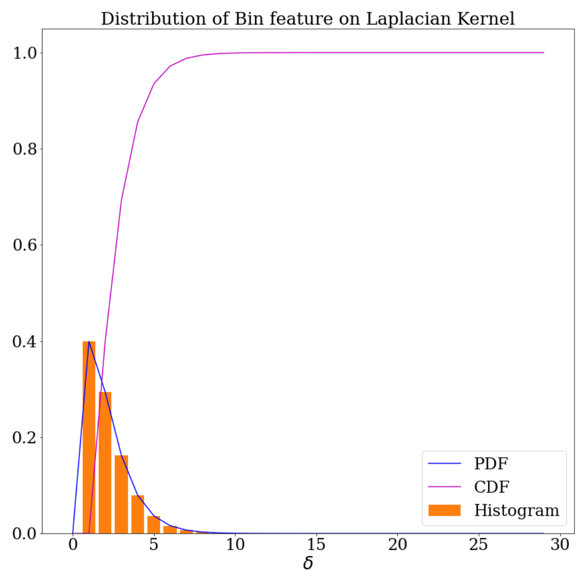
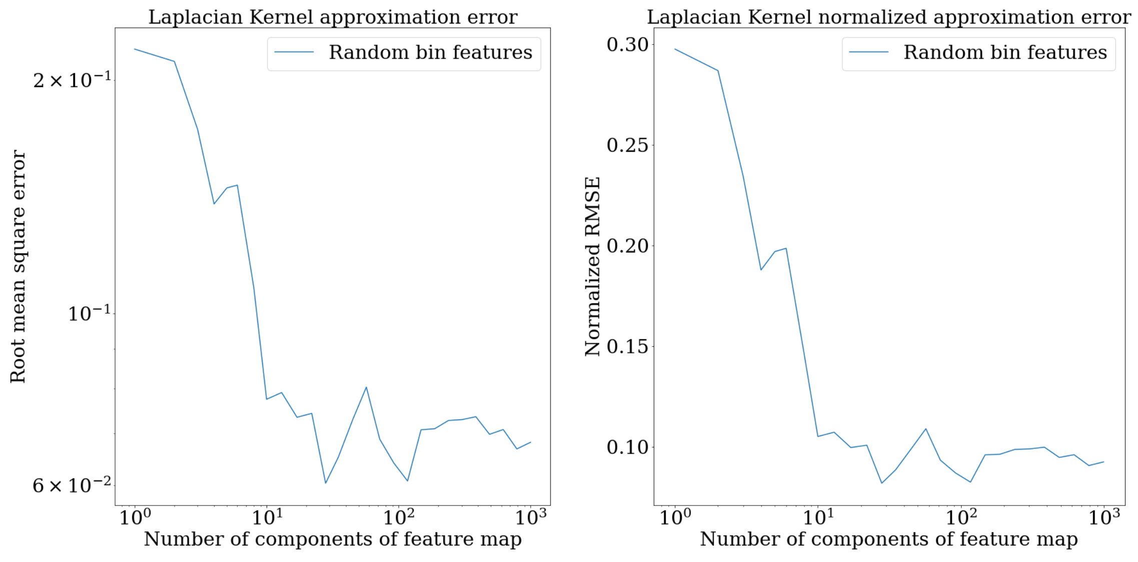


Figure 3.4.: This probability distribution $p(\delta)$ of Laplacian kernel is generated by the histogram of the derived formula in Algorithm 3. The derived distribution is sampled evenly on the interval $[0, 30]$ with 31 sampled values.



(a) RMSE (b) Normalized RMSE

Figure 3.5.: Approximation error of Laplacian Kernel value with random bin features.

Induced kernel

The induced kernel [17] is derived through decision stumps commonly used with the Adaboost algorithm. We first define the kernel k on $\mathcal{X} \times \mathcal{X}$ as

$$k(x, y) = \int_{\Theta} p(\omega)\phi(x, \theta)\phi(y, \theta)d\theta, \tag{3.11}$$

where the given function $\phi(x;\theta) : \mathcal{X} \times \Theta \to \mathbb{R}$ and probability distribution $p(\theta)$ is on Θ. By setting the form of the decision stump as $\phi(x;\theta) = sgn(x_i - t)$ with $\theta = (t, i)$, where t is a real number and $i \in [1, \dots, d]$ is some integer that indexes a component of $x \in \mathbb{R}^d$, we can have the following derivation,

$$\frac{1}{b-a}\int_a^b sgn(x-t)sgn(y-t)dt = 1 - 2\frac{y-x}{b-a}, \tag{3.12}$$

supposing t is selected uniformly at random on $[a, b]$ and $a \leq x < y \leq b$ for a one-dimensional case. A multi-dimensional induced kernel can be further given by

$$k(x, y) = 1 - \frac{1}{a}\|(x-y)\|_1. \tag{3.13}$$

by assuming t is sampled on $[-a, a]$ and $\mathcal{X}$ is contained in a hypercube $[-a, a]^d$.

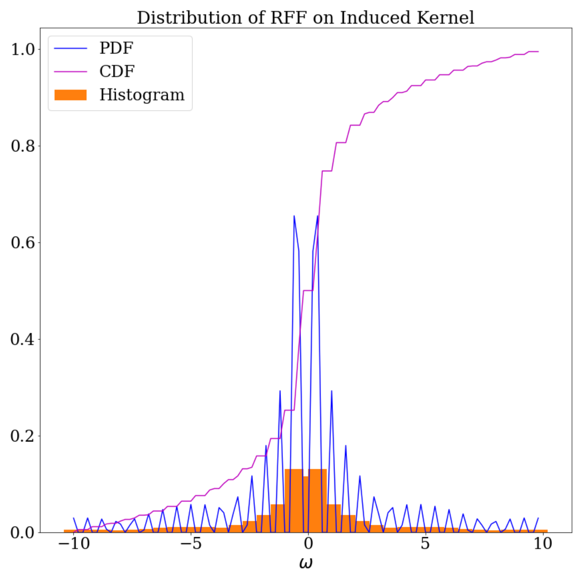
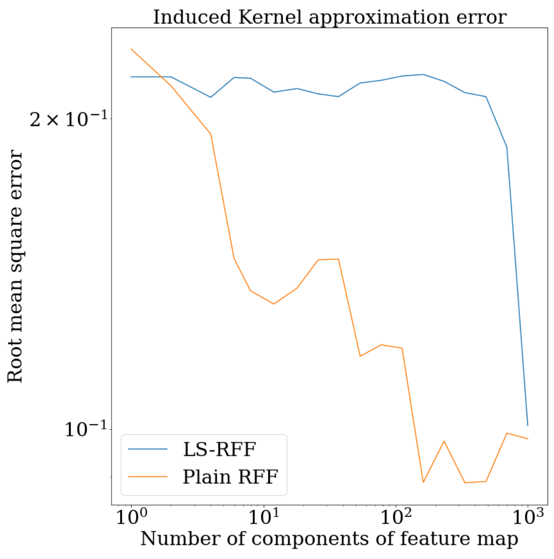
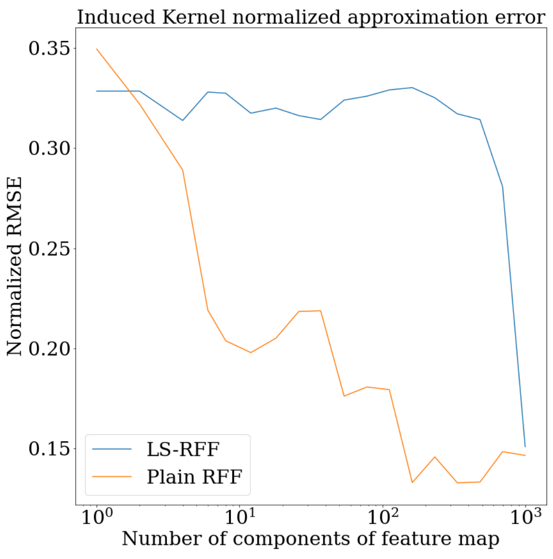


Figure 3.6.: This probability distribution $p(w)$ of Induced kernel is generated by the histogram of its Fourier transform.



(a) RMSE



(b) Normalized RMSE

Figure 3.7.: Approximation error of Induced Kernel value with random Fourier features.

Since the induced kernel does not have an explicit Fourier transform, the probability distribution in Figure 3.6 was generated using fixed-order Gaussian quadrature. The probability density of each sampled w was approximated by evaluating the definite integral of the convolution between $e^{-jw(x-y)}$ and induced kernel over the interval $[-10, 10]$. We sampled 101 values of w within the range $[-10, 10]$ using Gaussian quadrature with an order of 300. This numerical integration method allows us to approximate the probability distribution of the induced kernel efficiently and accurately.

Figure 3.7 illustrates the performance comparison between the Plain RFF and LS-RFF methods for approximating the induced kernel. The Plain RFF method consistently outperforms LS-RFF, except for a few initial components where LS-RFF shows slightly lower approximation errors. However, as the number of features increases, both methods converge to a similar NRMSE of around 15 percent. It is worth noting that the approximation accuracy saturates for both methods, indicating that further increasing the number of features does not significantly improve the accuracy. This result highlights the efficiency and effectiveness of the Plain RFF method for this particular approximation task.

Spline kernel

Spline kernel is usually considered to use for simulated experiments in other research as well [11] and is given by

$$k_{2r}(x,y) = 1 + \sum_{m>0}^{M} \frac{1}{m^{2r}} cos 2\pi m(x-y). \tag{3.14}$$

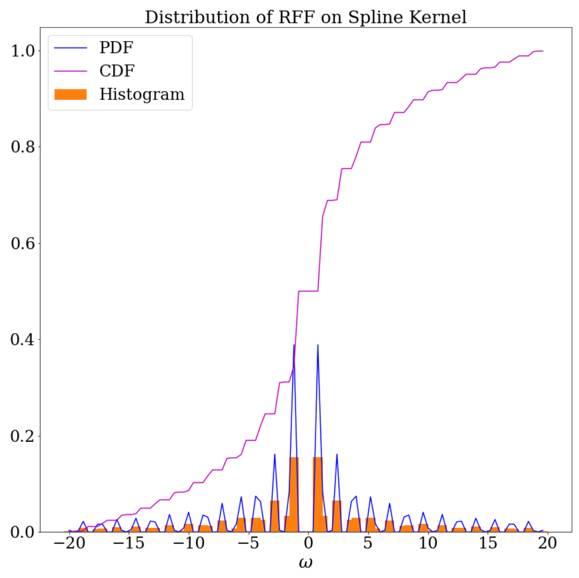
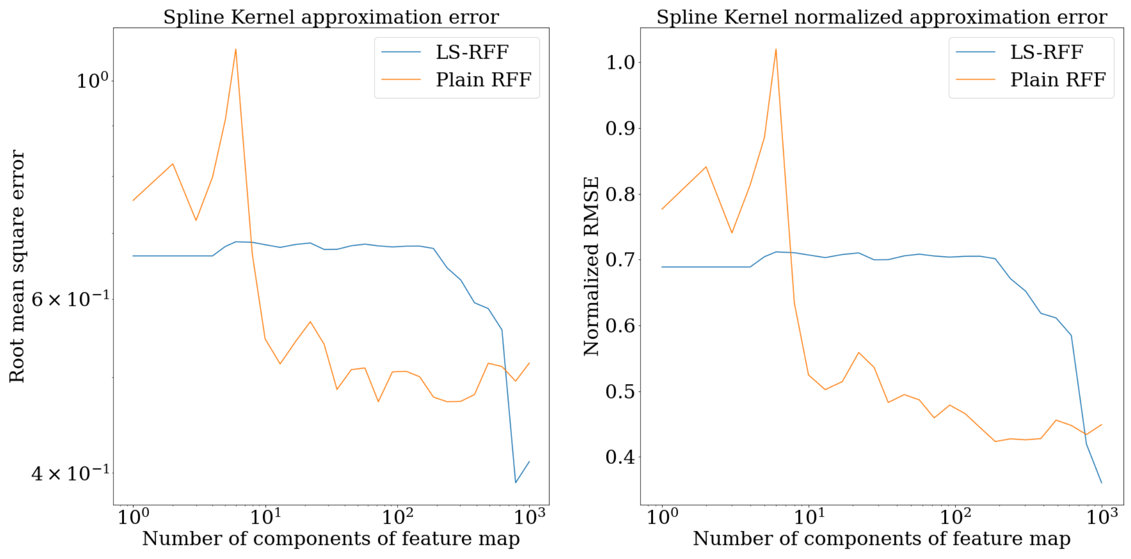


Figure 3.8.: This probability distribution $p(w)$ of Spline kernel is generated by the histogram of its Fourier transform.

The distribution shown in Figure 3.8 was approximated using Gaussian quadrature. The probability density of each sampled value of w was obtained by evaluating the definite integral of the convolution between $e^{-jw(x-y)}$ and the spline kernel over the interval $[-20, 20]$ with $r = 1$ and m from 1 to 10. To ensure accurate approximation, we sampled 101 values of w within the range $[-20, 20]$ using Gaussian quadrature with an order of 800. A finer sampling with 201 values of w has been tested as well; however, the result was worse. The choice of the order was based on the Nyquist-Shannon sampling theorem, which states that in order to accurately reconstruct a continuous-time signal from its samples, the sampling rate should be at least twice the highest frequency component present in the signal. Therefore, considering the range l and the highest frequency M, we need a total of $2lM$ points, which in this case was $2 \times (2 \times 20) \times 10 = 800$. This choice of order helps avoid aliasing and ensures the accurate representation of the distribution.

In Figure 3.9, LS-RFF initially demonstrates better performance compared to Plain RFF, which is a common observation. However, a notable difference is that LS-RFF achieves



(a) RMSE (b) Normalized RMSE

Figure 3.9.: Approximation error of Spline Kernel value with random Fourier features.

significantly better results toward the end of the approximation process. This finding suggests that the relationship between the frequency and offset may play a crucial role in determining the accuracy of the approximation. To further enhance the results, it is worth investigating the sampling strategy for the offsets b instead of uniformly sampling them from the interval $[0, 2\pi]$. By exploring different sampling strategies for b, we may be able to achieve even better approximation accuracy. While the results in Figure 3.9 exhibit a general trend of decreasing error, the approximation of the spline kernel highlights the limitations of random Fourier features in certain cases. Despite the decreasing error, it is evident that the approximation error saturates, indicating that the random Fourier features may struggle to capture the intricate characteristics of the spline kernel. This observation suggests that for kernels with more complex structures, alternative approaches or feature representations may be necessary to achieve higher approximation accuracy. It is possible that the generated probability distribution used to represent the relationship between the random Fourier features and the spline kernel may not be sufficient to create a feature map to capture the characteristics of the kernel.

The first challenge encountered when using random features to construct a feature map is determining the probability distribution for each feature. The choice of an appropriate probability distribution is crucial for the feature to accurately approximate the target kernel. For random Fourier features, the probability distribution is given by the Fourier transform of the kernel, while for bin features, it is defined by $\delta\ddot{k}(\delta)$. However, if the resulting distribution is not valid, such as not being positive everywhere, then the corresponding feature may not be suitable for approximating the kernel. In cases where there is no explicit form of the distribution function for a kernel-feature pair, numerical integration methods like Gaussian quadrature can be employed to sample the approximated distribution.

The experiments with the spline kernel highlight that not all kernels can be effectively approximated by random Fourier features, even when a probability distribution is generated. This emphasizes the need for considering alternative feature representations or methods when dealing with kernels that exhibit more complex structures. Otherwise, we have to refine the numerical scheme we use to increase the accuracy of the approximated distribution, which plays an important role in creating a quality feature map.

Furthermore, the comparison between different types of features suggests that when approximating the same kernel, the choice of feature depends on the trade-off between computational cost and accuracy, as demonstrated in the experiments with the Laplacian kernel. Random Fourier features show that utilizing the leverage score method can provide advantages in terms of accuracy with a potentially reduced number of components. However, in most cases, the accuracy of the leverage score method saturates early and was surpassed by the plain RFF method. In the case of the induced kernel, the plain RFF method even outperforms the leverage score method.

3.3.2. Convergence rate of excess risk on Kernel Ridge Regression

Kernel ridge regression involves computing the Gram matrix, which can be computationally expensive and memory-intensive, especially when dealing with large datasets. The Gram matrix is a pairwise similarity matrix that contains the inner products between all pairs of data points in the feature space induced by the kernel function.

The size of the Gram matrix grows quadratically with the number of data points, making it challenging to handle large-scale datasets. Computing and storing this matrix can consume a significant amount of memory resources, which can become a bottleneck for practical applications.

In order to address the computational challenges associated with kernel ridge regression, an effective approach is to approximate the feature map associated with the kernel and subsequently transform the training data points into the corresponding feature space. This transformation enables us to leverage linear ridge regression to learn the underlying relationship between the transformed data points and their corresponding targets. By doing so, we are able to preserve the desirable learning properties of nonlinear algorithms while significantly reducing the computational burden.

To demonstrate the ability of kernel approximation of random features, we referred to and modify the experiments of the simulated experiment [11] using spline kernel approximation, which is defined in Equation 3.14 with $r = 2$ and $M = 10$, in ridge regression and verified the convergence of excess risk at the same time. In this experiment, we demonstrated that the excess risk converged at a rate of $O(n^{-1/2})$, where n was the number of training data points, when the number of features was proportional to $d_{\boldsymbol{K}}^{\lambda}$, and λ was proportional to $n^{-1/2}$. This result was obtained using the leverage score RFF method.

The convergence rate of $O(n^{-1/2})$ indicated that as the number of training data points increased, the excess risk decreased at a slower rate, but still converged to a finite value. The data points X was sampled on $[0,1]$ and targets y was a Gaussian random variable with mean $f(x) = k_t(x, x_0)$ (for some $x_0 \in [0,1]$), variance $\sigma^2 = 0.01$, and $t = 3$. For the probability distribution $p(w)$ and feature map $z(\mathbf{x})$, we used the same algorithm as we used to approximate the spline kernel in the previous section. This was different from the referred experiment in [11], where $p(w)$ was uniform on $[0,1]$ and $z(\mathbf{x}) = k_r(w, x)$. The generated dataset was divided into a training set and a testing set using a ratio of 2:1. We conducted experiments with varying values of x_0 and obtained the following results.

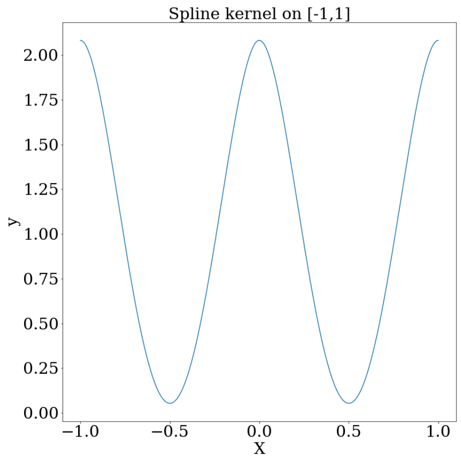
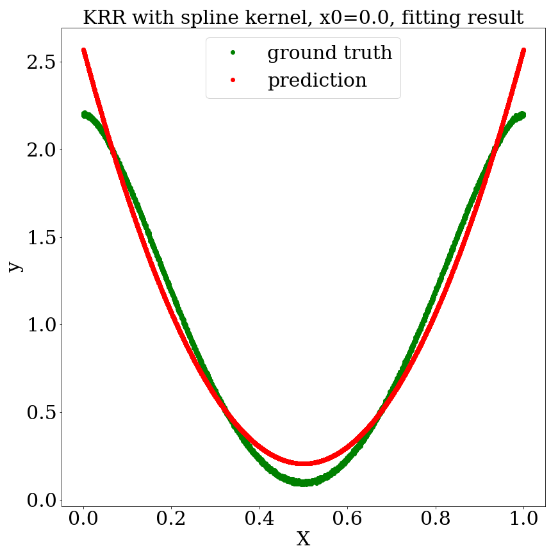


Figure 3.10.: The plot of the spline kernel used as the target values y.

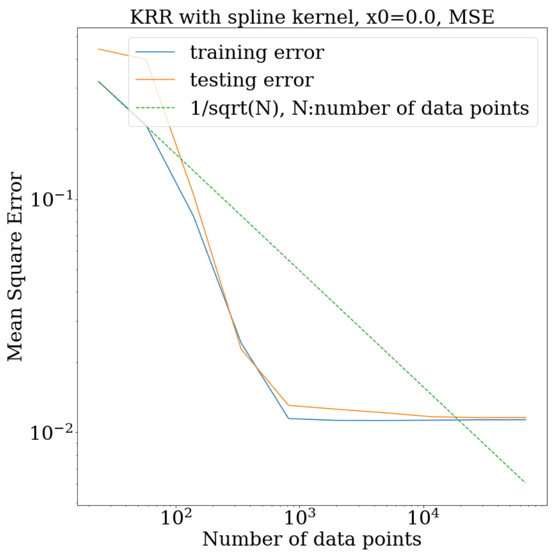
Figure 3.10 depicts the spline kernel on the interval $[-1,1]$ using the same sampling scheme as in Figure 3.8. This sampling scheme is employed to prevent aliasing and ensure the accurate representation of the complete spline kernel. Since the range of the input variable X is constrained to $[0,1]$ and the shift parameter x_0 takes values in the interval $[0,1]$, it can be interpreted as sliding a window of width 1 over the spline kernel and shifting it to the left based on the value of x_0. This allows us to capture a portion of the spline kernel as illustrated in Figure 3.10.

In Figure 3.11, we observe the right part of the spline kernel in Figure 3.10, specifically the interval $[0,1]$, which exhibits a convex curve. The predicted results closely resemble the ground truth in many cases, particularly in the linear portion of the convex curve. The convergence rate of the risk aligns with the theoretical expectation until training with approximately 1000 data points. This suggests that the number of features employed in the model might not be sufficient to capture the complexity of the data distribution.

In Figure 3.12, we examine both parts of the spline kernel displayed in Figure 3.10, focusing on the interval $[-0.2, 0.8]$, where a cubic curve is observed. It becomes apparent that the feature map is inadequate for accurately representing a dataset in this form. Interestingly,

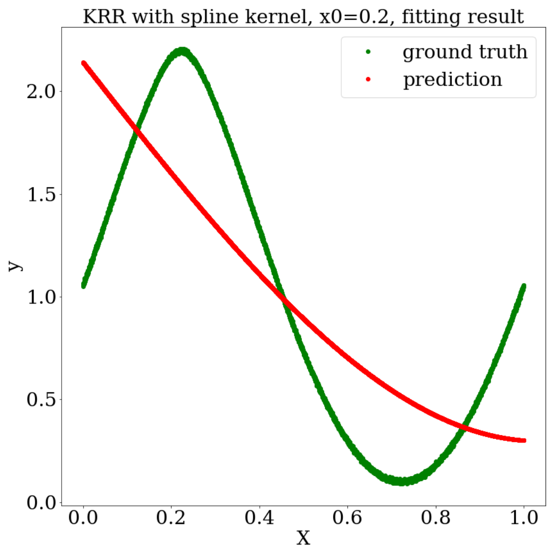


(a) Fitting result with shift $x_0 = 0$.

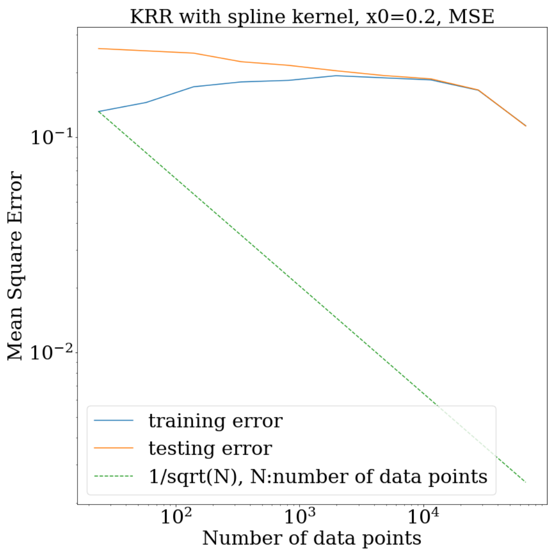


(b) Fitting error with shift $x_0 = 0$.

Figure 3.11.: Kernel ridge regression results using approximated feature map of spline kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0$.

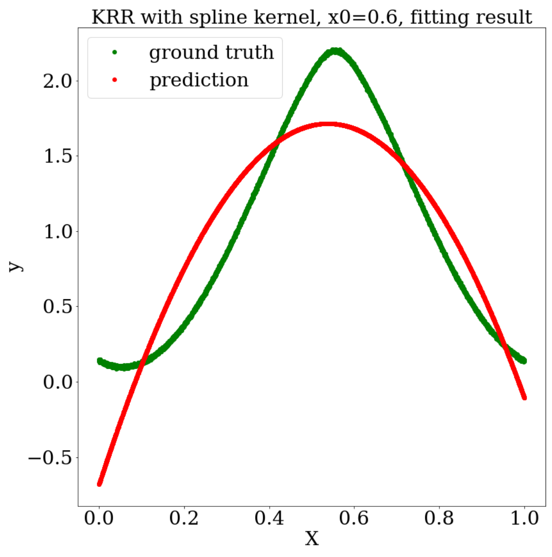


(a) Fitting result with shift $x_0 = 0.2$.

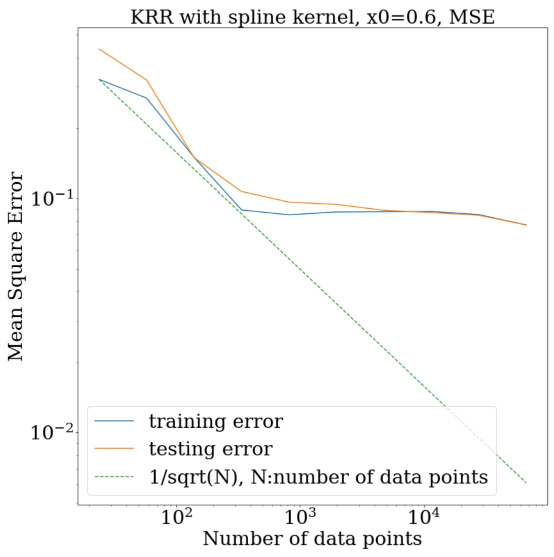


(b) Fitting error with shift $x_0 = 0.2$.

Figure 3.12.: Kernel ridge regression results using approximated feature map of spline kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.2$.



(a) Fitting result with shift $x_0 = 0.6$.



(b) Fitting error with shift $x_0 = 0.6$.

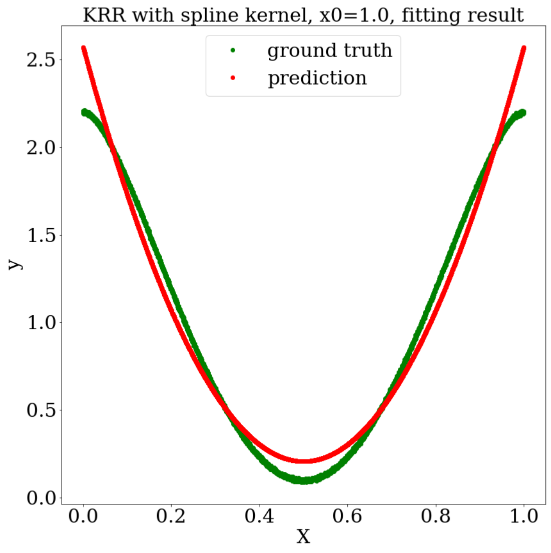
Figure 3.13.: Kernel ridge regression results using approximated feature map of spline kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.6$.

the convergence rate does not decrease as the number of training data points increases; instead, it even shows an increasing trend. This further underscores the limitations of the feature map in capturing the underlying relationship in this particular scenario.

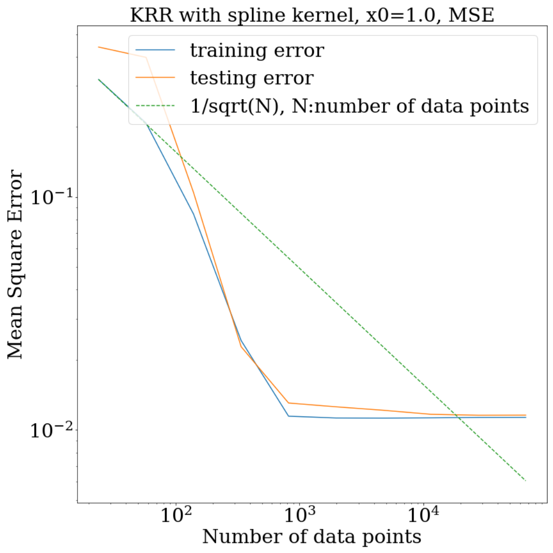
In Figure 3.13, we analyze a significant portion of the left part of the spline kernel depicted in Figure 3.10, specifically focusing on the interval $[-0.6, 0.4]$, which exhibits a concave curve. It is observed that the feature map is relatively more suitable for approximating this particular data form compared to the cubic curve case. However, the approximation is still not accurate enough to capture the true underlying relationship. The convergence rate of the risk aligns with the theoretical expectation until training with approximately 300 data points, suggesting that the number of features employed in the model may not be sufficient to adequately capture the complexity of the data distribution with the given dataset size.

In Figure 3.14, we analyze the left part of the spline kernel depicted in Figure 3.10, specifically focusing on the interval $[-1.0, 0.0]$, which exhibits a convex curve similar to Figure 3.11. The convergence rate of the risk aligns with the theoretical expectation until training with approximately 1000 data points, yielding similar results as observed in Figure 3.11.

In the simulated ridge regression experiments, the convergence of the excess risk was highly dependent on the choice of shift parameter, indicating that the feature map had limitations in accurately capturing the underlying relationship across different datasets. The performance of the feature map in terms of risk reduction may vary significantly de-



(a) Fitting result with shift $x_0 = 1.0$.



(b) Fitting error with shift $x_0 = 1.0$.

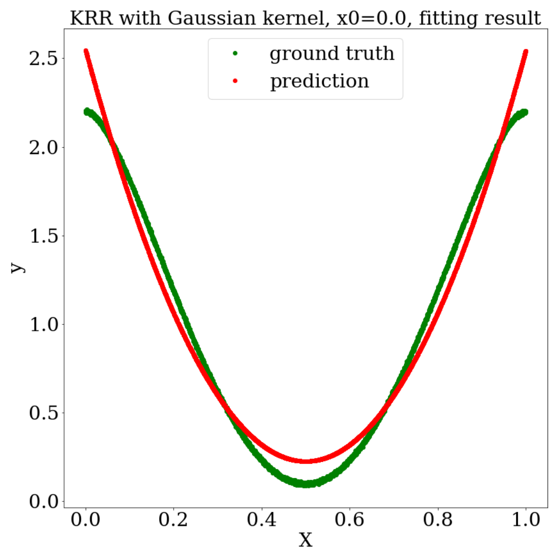
Figure 3.14.: Kernel ridge regression results using approximated feature map of spline kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 1.0$.

pending on the specific shift value used. This observation highlights the limitations of the feature map design in Algorithm 2 ineffectively capturing the complex patterns and dynamics present in diverse datasets. As a result, the performance in terms of risk convergence may be unsatisfactory. To address this issue, it is necessary to either modify the design of the feature map based on prior analysis of the dataset or increase the number of random Fourier features to enhance the representation capabilities of the model.

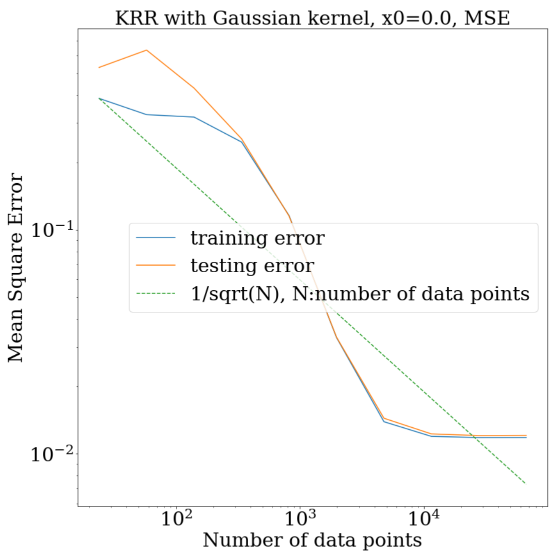
In Figure 3.15, Figure 3.16, and Figure 3.17, we present the best results obtained by varying the shifting parameter for different kernels. The results for other parameter values can be found in Appendix A.1. It can be observed that the excess risk saturates at around 0.01 for all the kernels at certain points.

In terms of convergence rate, the spline kernel demonstrated the best performance compared to the other kernels. Surprisingly, the Gaussian kernel, which is typically popular and widely used, exhibits the worst performance. It required approximately 10,000 data points to achieve the lowest possible error, while the Laplacian kernel and induced kernel required only half that amount. Remarkably, the spline kernel required even fewer data points, less than 1000, to reach optimal accuracy.

The choice of the kernel can have a significant impact, and determining the most suitable kernel for different scenarios can be explored in a chapter discussing this technique in [7].

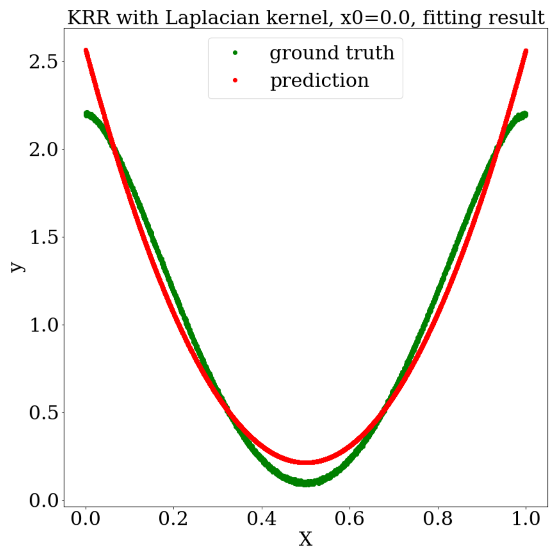


(a) Fitting result with shift $x_0 = 0.0$.

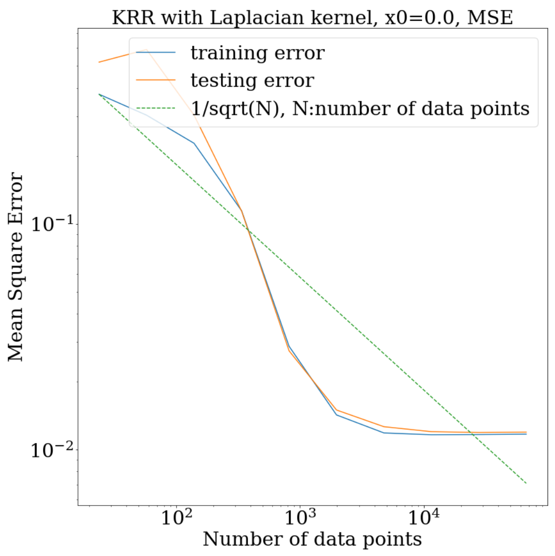


(b) Fitting error with shift $x_0 = 0.0$.

Figure 3.15.: Kernel ridge regression results using approximated feature map of Gaussian kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.0$.

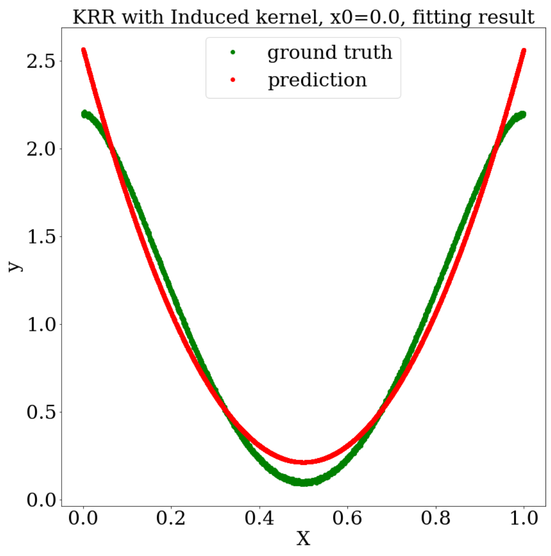


(a) Fitting result with shift $x_0 = 0.0$.

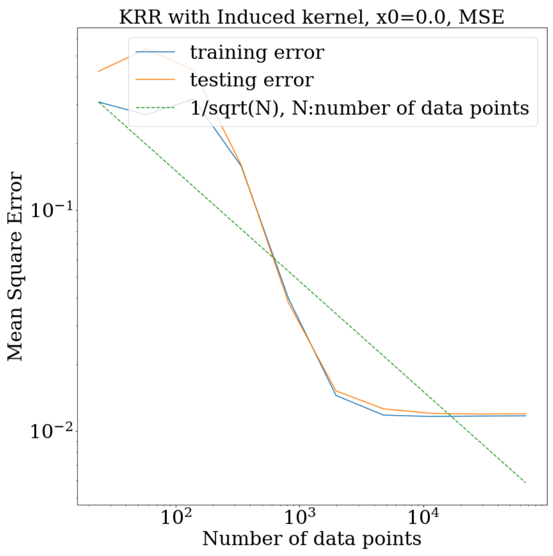


(b) Fitting error with shift $x_0 = 0.0$.

Figure 3.16.: Kernel ridge regression results using approximated feature map of Laplacian kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.0$.



(a) Fitting result with shift $x_0 = 0.0$.



(b) Fitting error with shift $x_0 = 0.0$.

Figure 3.17.: Kernel ridge regression results using approximated feature map of Induced kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.0$.

3.3.3. Performance of the random feature approximation

To assess the efficiency improvement gained from utilizing random features for approximating kernels in learning algorithms, a comparison was made between the feature map of each kernel and the approximated kernel obtained by training on the same dataset using Kernel Ridge Regression. In order to demonstrate this improvement, a small dataset was chosen from scikit-learn called ”Diabetes” [4], as it contains a manageable number of samples compared to the resource-intensive ”California housing” dataset. The ”Diabetes” dataset consists of 442 samples, where each sample consists of 10 independent variables ranging from -0.2 to 0.2, along with a dependent variable ranging from 25 to 346.

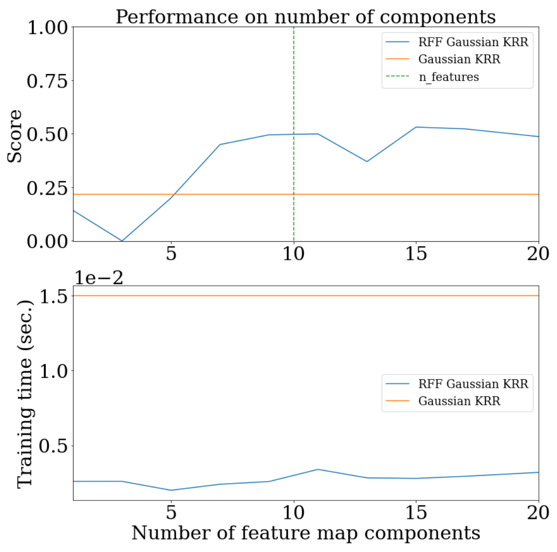
In order to perform learning with feature maps, the input data needs to be transformed before being used as input for Linear Ridge Regression. In this study, plain Random Fourier Features (RFF) were employed for each kernel, with the maximum number of components limited to approximately 5% of the total number of data points. This restriction ensures that the approximation is applied while conserving storage space. It is important to note that the Kernel Ridge Regression (KRR) implementation in scikit-learn only supports certain metrics for popular kernels such as the Gaussian and Laplacian kernels. For other kernels like the induced kernel and spline kernel, a precomputed kernel matrix must be provided to the API.

To measure the computational time of each algorithm, the processes were repeated five times under the same settings, and the average duration was recorded. The timing of the random feature approximation process encompasses the generation of random variates for the features, the transformation of the original input into the feature space, and the fitting of a linear Ridge Regression model. As for Kernel Ridge Regression, the time spent by the ”fit” API in generating the model was measured. This comprehensive timing analysis allows for a thorough understanding of the computational costs associated with both the random feature approximation and Kernel Ridge Regression approaches. The experimental results are presented in three plots to analyze the relationship between the number of components in the feature map, the R^2 score, and the computational time.

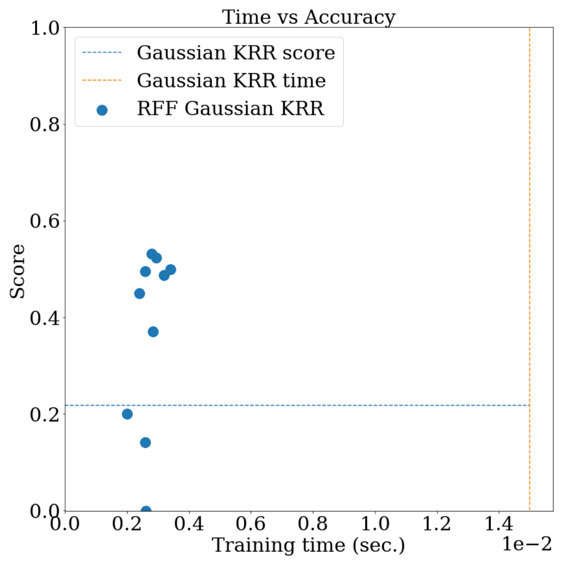
The first plot illustrates how increasing the number of components in the feature map affects the R^2 score. The dimensionality of the original data is also specified to examine the impact of the dimensionality of the representation of the data, which depends on the number of components of the feature map. The second plot displays the training time usage of both the random feature approximation and Kernel Ridge Regression algorithms. This comparison allows for a straightforward understanding of the computational efficiency of each method. In the last plot, the trade-off between accuracy and time is analyzed. It compares the training time and score of the two algorithms, dividing the plot into four regions using two lines. The upper-left region represents cases where the approximated feature map is highly efficient, surpassing Kernel Ridge Regression in terms of both score and training time. Conversely, the lower-right region indicates cases where the approximated feature map yields worse results than Kernel Ridge Regression and requires more time for training. The remaining part of the plot demonstrates that better results can be achieved with increased training time, which is not the primary concern of this experiment.

Figure 3.18 demonstrates that the score achieved by using KRR with the Gaussian kernel is relatively low, around 0.25. However, when employing the RFF approximation for Gaussian KRR, the score surpasses that of KRR with just a few components, even fewer than the original number of features (dimensionality) of the data.

Both methods exhibited fast training times in this case, with the approximated KRR being approximately three times faster than the original KRR. Furthermore, the time usage slightly increased as more features were used. Consequently, the trade-off between time and accuracy was negligible, as using only 20 components could yield a higher accuracy score of approximately 0.5 while saving training time.



(a) Relationship between the number of components used in the feature map, computational time, and accuracy.

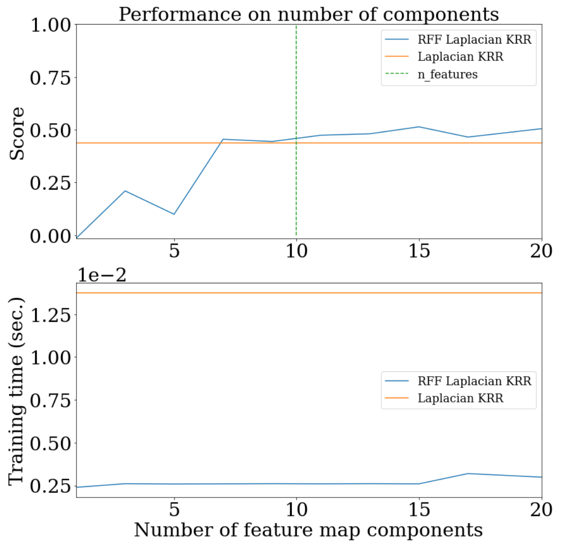


(b) Trade-off between computational time and accuracy.

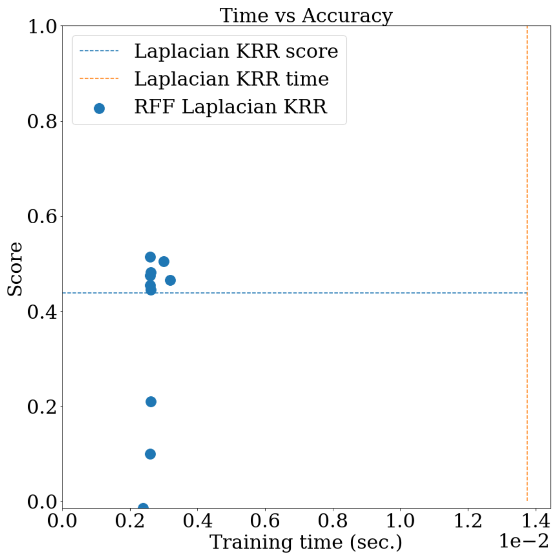
Figure 3.18.: Performance comparison of Kernel Ridge Regression using Gaussian kernel and Ridge Regression using approximated feature map.

Figure 3.19 illustrates that the score achieved by using KRR with the Laplacian kernel is 0.44. After employing the RFF approximation for Laplacian KRR with only 7 components, the score improves slightly and reaches around 0.5. Notably, the time usage of the approximated KRR is significantly less than that of the original KRR. As a result, the trade-off between time and accuracy was not a significant concern in this case, as the approximated KRR achieved a similar score to the original KRR while requiring less training time.

In Figure 3.20, it is observed that the score achieved by using KRR with the induced kernel is 0.34. However, after employing the RFF approximation for induced KRR with only 7 components, the score improves significantly to over 0.4. Eventually, with a larger



(a) Relationship between the number of components used in the feature map, computational time, and accuracy.



(b) Trade-off between computational time and accuracy.

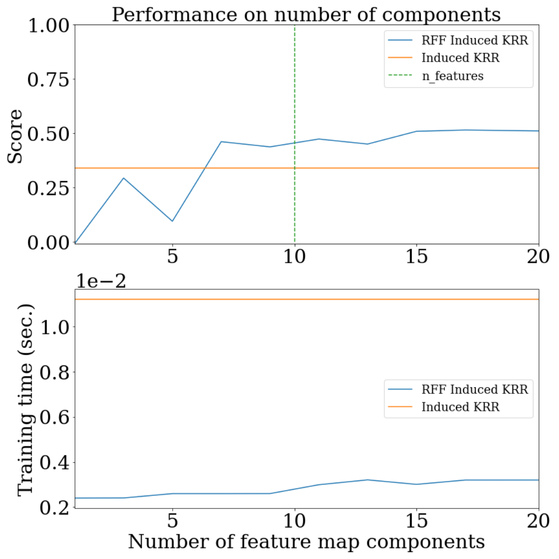
Figure 3.19.: Performance comparison of Kernel Ridge Regression using Laplacian kernel and Ridge Regression using approximated feature map.

number of components, the score reaches around 0.5. The time usage of the approximated KRR is around one-fourth of the time required by the original KRR. This indicates that in terms of the trade-off between time and accuracy, the RFF approximation is a superior learning algorithm in this particular case.

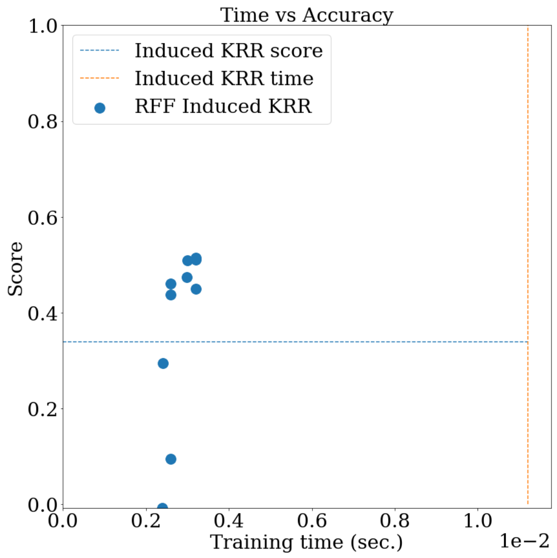
Since the spline kernel was not defined for multivariate [11], here we used L2-norm to calculate. In Figure 3.21, it is observed that for the multivariate spline kernel, KRR achieves a higher score compared to the RFF approximation. However, there is an exception when using 17 components for the RFF approximation, where the score briefly surpasses that of KRR. In terms of time usage, the RFF approximation requires less training time compared to the original KRR. However, the trade-off between time and accuracy was not favorable in this case. To achieve a higher score with the RFF approximation, it was necessary to increase the number of components and sacrifice storage space.

In summary, the performance experiment demonstrated that the RFF-approximated KRR generally outperforms the original KRR. Among the kernels used in KRR, the spline kernel achieved the highest score, while the Gaussian kernel performed the worst. However, when using the RFF approximation, the Laplacian kernel showed the best performance, while the spline kernel performed relatively worse.

In terms of efficiency, the RFF approximation significantly improved the performance of the Gaussian kernel and induced kernel compared to their respective original KRR algo-



(a) Relationship between the number of components used in the feature map, computational time, and accuracy.

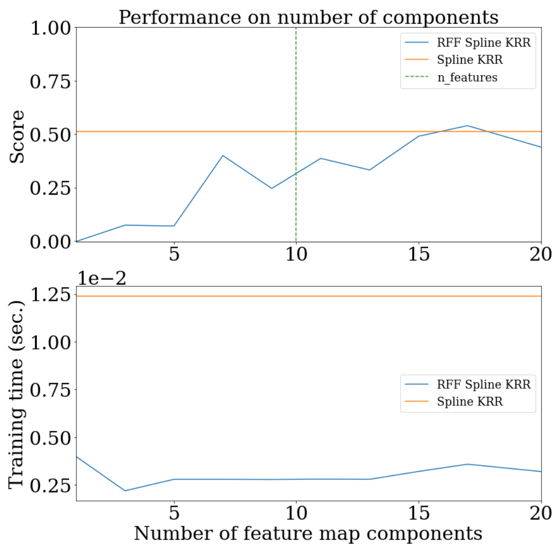


(b) Trade-off between computational time and accuracy.

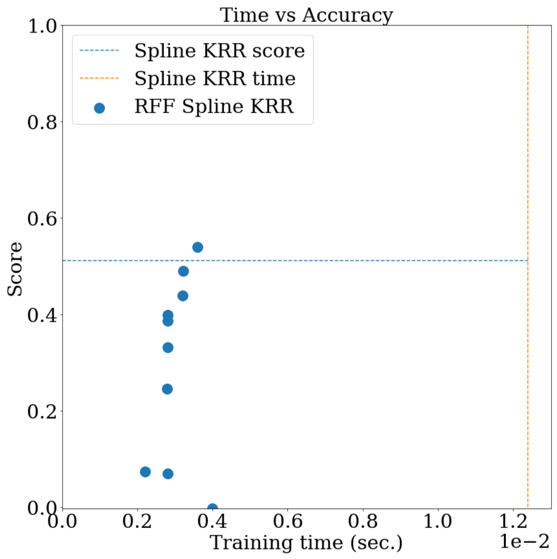
Figure 3.20.: Performance comparison of Kernel Ridge Regression using Induced kernel and Ridge Regression using approximated feature map.

rithms. This suggests that the RFF approximation can provide substantial computational advantages without sacrificing accuracy.

In addition, since the score of both methods was relatively low, the choice of using the ”Diabetes” dataset might not be the best. Dataset plays a crucial role in evaluating the performance and generalization of learning algorithms. To enhance confidence in the experimental results, it is suggested to select a larger dataset with more informative features, as long as the storage space can accommodate it. Considering benchmark datasets specifically designed for testing purposes can be an option. It offers standardized evaluation metrics and established baselines for better comparisons with existing methods. Besides, opting for a dataset that is representative, diverse, and better aligned with the problem domain, the evaluation becomes more comprehensive.



(a) Relationship between the number of components used in the feature map, computational time, and accuracy.



(b) Trade-off between computational time and accuracy.

Figure 3.21.: Performance comparison of Kernel Ridge Regression using Spline kernel and Ridge Regression using approximated feature map.

4. Conclusion

In this thesis, we address the scaling issue of kernel methods in machine learning algorithms when applied to larger datasets. To overcome this problem, a kernel approximation method using random features is proposed. The effectiveness of this method is demonstrated through the approximation of kernel values, showing its capability to replace the computationally demanding kernel methods. Moreover, the convergence of the excess risk is empirically and theoretically verified, highlighting the trustworthiness and utility of the kernel approximation method, particularly in the context of Kernel Ridge Regression.

This thesis begins by explaining the application of kernel methods in algorithms such as Kernel Ridge Regression and Support Vector Machines, highlighting their potential for capturing complex patterns and structures in data. To address the computational challenges associated with these methods, an efficient approximation technique utilizing random features is introduced. The thesis provides a detailed description of the implementation process, including the generation of probability distributions for random feature parameters. It presents both random Fourier features and random bin features, along with algorithms for constructing specific feature maps suitable for kernel approximation tasks.

The experiments conducted with the spline kernel underscore the fact that not all kernels can be accurately approximated using random Fourier features, even when a probability distribution is generated. This highlights the importance of exploring alternative feature representations or methods when dealing with kernels that possess more intricate structures. Additionally, the comparison between different types of features reveals that the choice of feature depends on the trade-off between computational cost and accuracy. Random Fourier features, specifically the leverage score method, offer advantages in terms of accuracy with a potentially reduced number of components. However, the accuracy of the leverage score method often saturates early and is surpassed by the plain RFF method. These findings emphasize the need to carefully select the appropriate feature representation and consider the computational resources available to achieve the desired balance between accuracy and efficiency in kernel approximation tasks.

The simulated ridge regression experiments reveal that the convergence of the excess risk is heavily influenced by the choice of the shift parameter, indicating limitations in the feature map's ability to accurately capture the underlying relationships across different datasets. The performance of the feature map in terms of risk reduction varies significantly depending on the specific shift value used, highlighting the inadequacy of the feature map design in capturing the complexity of diverse datasets. To overcome this limitation, potential solutions include modifying the feature map design based on dataset analysis or increasing the number of random Fourier features to improve the representation capabilities of models and enhance risk convergence.

The performance experiment conducted in this study highlighted the superiority of the RFF-approximated KRR compared to the original KRR. The RFF-approximated KRR consistently outperformed its original counterpart across different kernels. The spline kernel achieved the highest score in KRR, while the Gaussian kernel performed the worst. However, when utilizing the RFF approximation, the Laplacian kernel exhibited the best performance, while the spline kernel performed relatively worse. In terms of efficiency, the RFF approximation proved to be highly beneficial for the Gaussian and induced kernels, significantly enhancing computational efficiency without compromising accuracy.

To gain deeper insights, it is valuable to explore the performance of different feature maps using a well-performing kernel and investigate the impact of approximation schemes. For example, one can examine the Laplacian kernel with bin features or random Fourier features and compare their performance in learning tasks. Even when using the same kernel, different feature maps can lead to significant variations in learning outcomes, especially when dealing with large-scale datasets. Additionally, it is recommended to extend the experiments beyond Kernel Ridge Regression and includes a classification problem using Kernel Support Vector Machines. This allows for testing the effectiveness of feature maps across various learning algorithms and problem domains. Lastly, it is advisable to choose a larger and benchmark dataset for the experiments, as opposed to relying solely on simulated or provided datasets like "Diabetes" from Scikit-learn. This ensures the generalizability of the findings and provides a more realistic evaluation of the performance of kernel methods and their feature maps.

To sum up, this thesis provides a clear introduction to understanding the mechanism of random features approximation of covariance kernels and demonstrates the capability of this method with step-by-step implementation details, kernel value approximation, and Kernel Ridge Regression experiments. Apart from that, it provides directions for future researchers to extend the work.

Bibliography

[1] Haim Avron, Michael Kapralov, Cameron Musco, Christopher Musco, Ameya Velingker, and Amir Zandieh. Random fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *International conference on machine learning*, pages 253–262. PMLR, 2017.

[2] Francis Bach. On the equivalence between kernel quadrature rules and random feature expansions. *The Journal of Machine Learning Research*, 18(1):714–751, 2017.

[3] Aude Billard. Support vector machines, applied machine learning micro-455, epfl, 2020.

[4] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[5] CodeEmporium. The kernel trick - the math you should know!, 2018.

[6] Andrew Cotter, Joseph Keshet, and Nathan Srebro. Explicit approximations of the gaussian kernel. *arXiv preprint arXiv:1109.4603*, 2011.

[7] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014.

[8] Charles R. Harris, K. Jarrod Millman, Stéfan J van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585:357–362, 2020.

[9] Thomas Hofmann, Bernhard Schölkopf, and Alexander J. Smola. Kernel methods in machine learning. *The Annals of Statistics*, 36(3):1171 – 1220, 2008.

[10] Cho-Jui Hsieh, Si Si, and Inderjit Dhillon. A divide-and-conquer solver for kernel support vector machines. In *International conference on machine learning*, pages 566–574. PMLR, 2014.

[11] Zhu Li, Jean-Francois Ton, Dino Oglic, and Dino Sejdinovic. Towards a unified analysis of random fourier features. *The Journal of Machine Learning Research*, 22(1):4887–4937, 2021.

[12] Fanghui Liu, Xiaolin Huang, Yudong Chen, and Johan AK Suykens. Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7128–7148, 2021.

[13] Fanghui Liu, Xiaolin Huang, Yudong Chen, Jie Yang, and Johan Suykens. Random fourier features via fast surrogate leverage weighted sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4844–4851, 2020.

[14] Ulrike von Luxburg. Statistical machine learning - kernel methods for supervised learning, 2022.

[15] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[16] Ali Rahimi and Benjamin Recht. Random features for large-scale kernel machines. *Advances in neural information processing systems*, 20, 2007.

[17] Ali Rahimi and Benjamin Recht. Uniform approximation of functions with random bases. In *2008 46th Annual Allerton Conference on Communication, Control, and Computing*, pages 555–561. IEEE, 2008.

[18] Ali Rahimi and Benjamin Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. *Advances in neural information processing systems*, 21, 2008.

[19] Alexander J Smola. Sparse greedy matrix approximation for machine learning. In *Proceedings of the 17th international conference on machine learning, June 29-July 2 2000*. Morgan Kaufmann, 2000.

[20] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

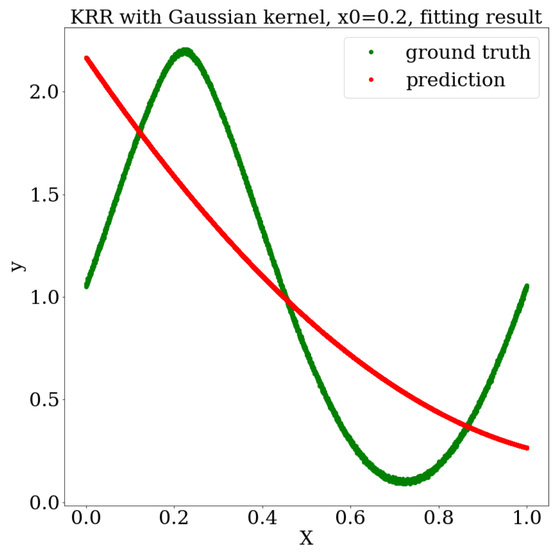
[21] Max Welling. Kernel ridge regression note of introduction to machine learning, 2019.

[22] Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems*, 13, 2000.

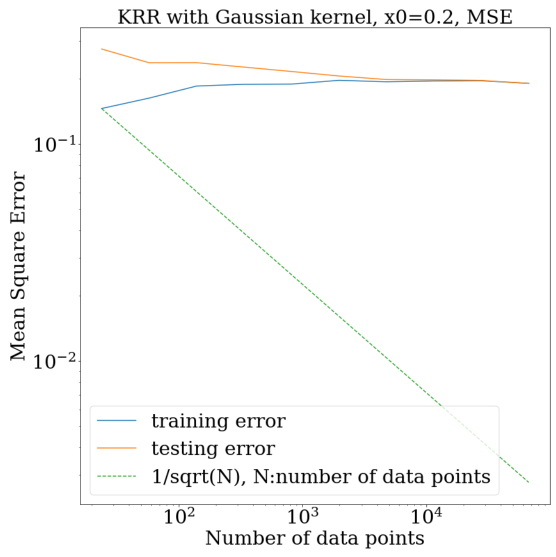
A. Appendix

A.1. KRR convergence rate of excess risk

A.1.1. Gaussian kernel

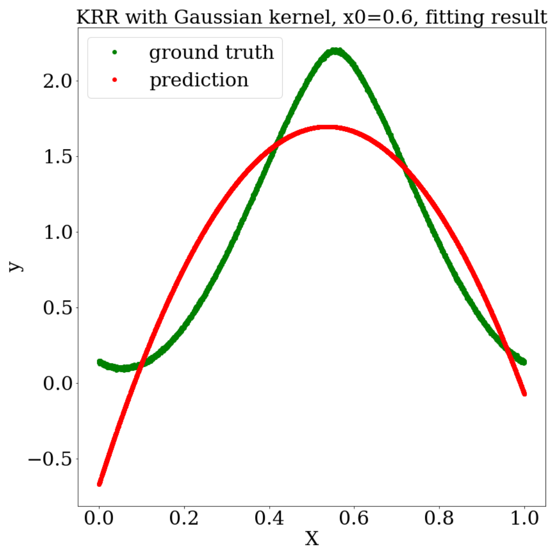


(a) Fitting result with shift $x_0 = 0.2$.

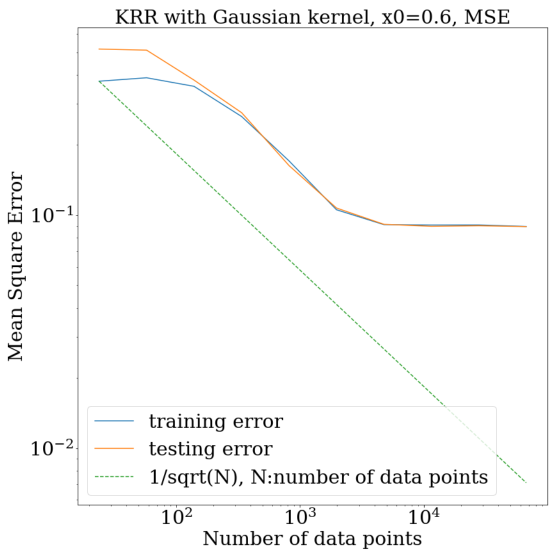


(b) Fitting error with shift $x_0 = 0.2$.

Figure A.1.: Kernel ridge regression results using approximated feature map of Gaussian kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.2$.



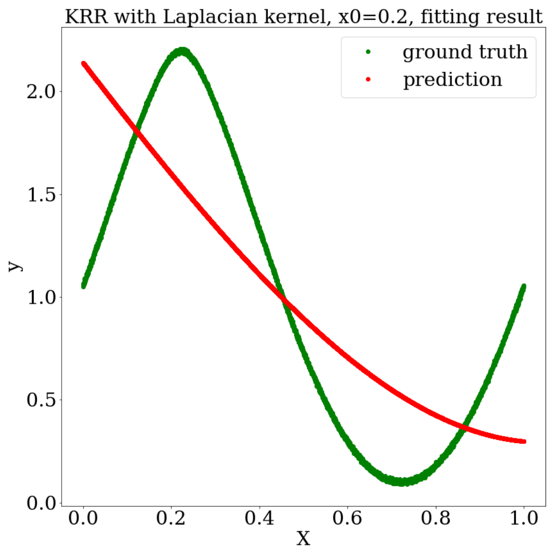
(a) Fitting result with shift $x_0 = 0.6$.



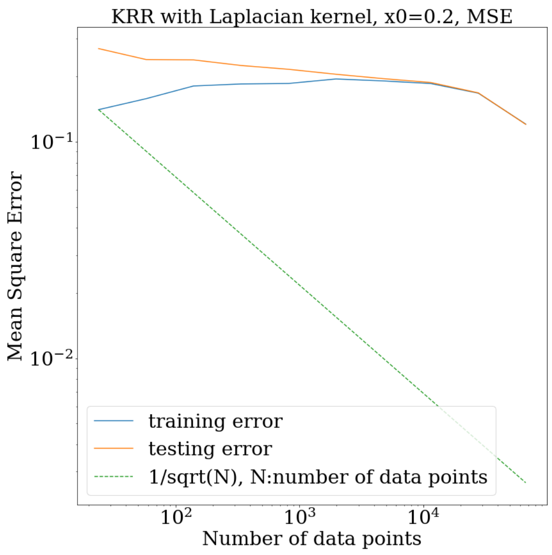
(b) Fitting error with shift $x_0 = 0.6$.

Figure A.2.: Kernel ridge regression results using approximated feature map of Gaussian kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.6$.

A.1.2. Laplacian kernel

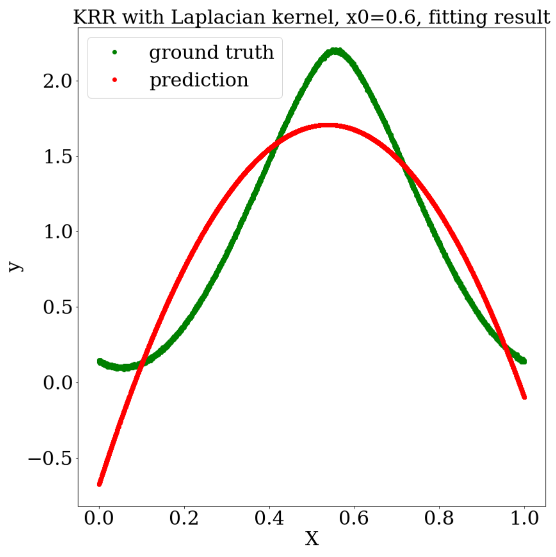


(a) Fitting result with shift $x_0 = 0.2$.

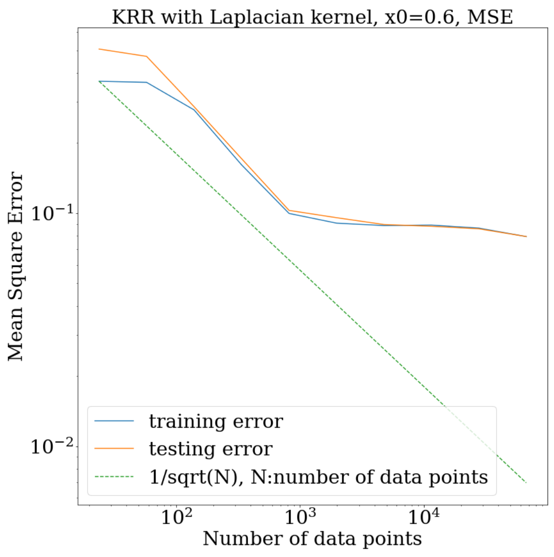


(b) Fitting error with shift $x_0 = 0.2$.

Figure A.3.: Kernel ridge regression results using approximated feature map of Laplacian kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.2$.



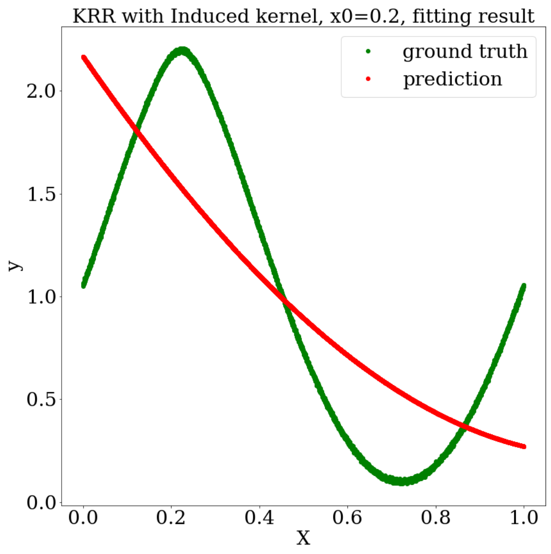
(a) Fitting result with shift $x_0 = 0.6$.



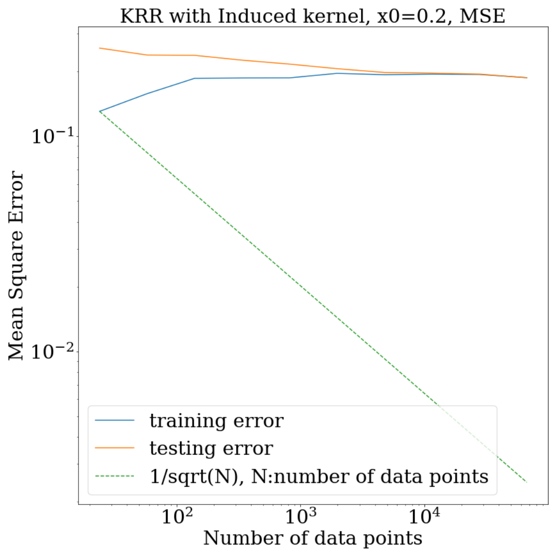
(b) Fitting error with shift $x_0 = 0.6$.

Figure A.4.: Kernel ridge regression results using approximated feature map of Laplacian kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.6$.

A.1.3. Induced kernel

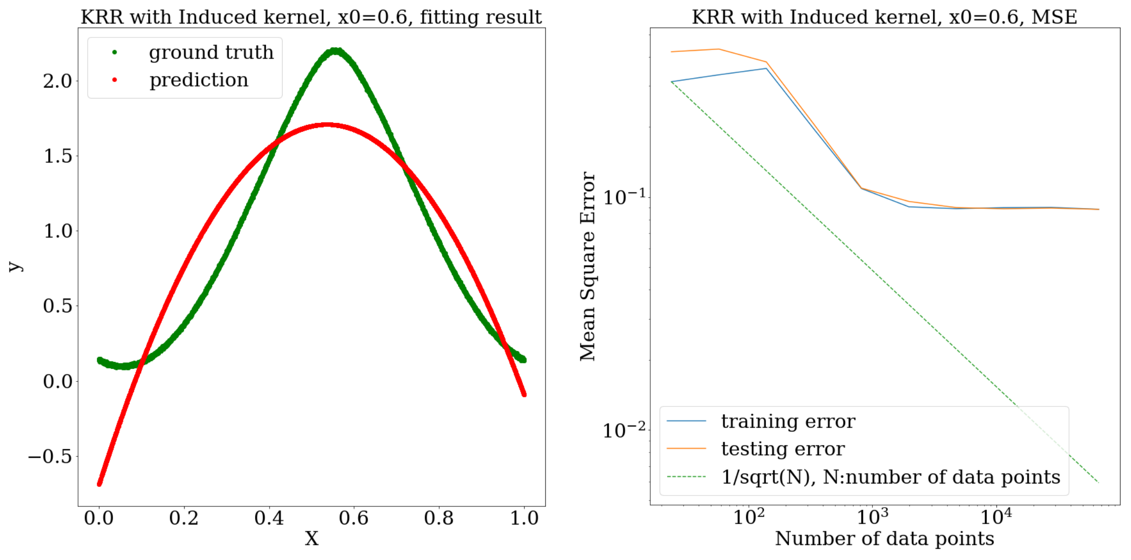


(a) Fitting result with shift $x_0 = 0.2$.



(b) Fitting error with shift $x_0 = 0.2$.

Figure A.5.: Kernel ridge regression results using approximated feature map of Induced kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.2$.



(a) Fitting result with shift $x_0 = 0.6$.

(b) Fitting error with shift $x_0 = 0.6$.

Figure A.6.: Kernel ridge regression results using approximated feature map of Induced kernel with the LS-RFF method. The fitting target y is shifted by $x_0 = 0.6$.