# How Controlled Features in Random Fourier Features Perform?

**Wang Ma** *
Department of Statistics and Data Science
Southern University of Science and Technology
maw2020@mail.sustech.edu.cn

**Qiang Hu** *
Department of Statistics and Data Science
Southern University of Science and Technology
12111214@mail.sustech.edu.cn

**Junjie Qiu** *
Department of Statistics and Data Science
Southern University of Science and Technology
12111831@mail.sustech.edu.cn

**Jingjuan Huang** *
Department of Statistics and Data Science
Southern University of Science and Technology
12112847@mail.sustech.edu.cn

## Abstract

Random Fourier Features (RFF) is a method to approximate the kernel function $k(\cdot, \cdot)$ by a random feature map $\phi(\cdot)$. It is considered as a breakthrough in kernel methods, since it makes kernel methods scalable to large datasets. It is widely used in kernel methods, such as Support Vector Machine (SVM), Gaussian Process Regression (GPR), etc. In this paper, we conduct experiments to investigate the effect of dimensionality and sampling distribution on the performance of RFF. We also compare the performance of RFF with other kernel methods, such as Support Vector Machine (SVM), Gaussian Process Regression (GPR), etc. We find that RFF is sensitive to the dimensionality and sampling distribution. We also find that RFF is comparable to other kernel methods in terms of accuracy, but it is much faster than other kernel methods in terms of training and inference time. Interestingly, we also find that RFF is comparable to other kernel methods in terms of interpretability. Our code is available on GiHub Repo.

## 1 Introduction

Kernel methods are widely used in machine learning, such as Support Vector Machine (SVM), Gaussian Process Regression (GPR), etc. However, kernel methods are not scalable to large datasets, since they require computing the kernel function $k(\cdot, \cdot)$ for all pairs of data points. This requires $O(n^2)$ time and $O(n^2)$ space, where $n$ is the number of data points. This makes kernel methods impractical for large datasets.

Random Fourier Features (RFF) is a method to approximate the kernel function $k(\cdot, \cdot)$ by a random feature map $\phi(\cdot)$. It is considered as a breakthrough in kernel methods, since it makes kernel methods scalable to large datasets. It is widely used in kernel methods, such as Support Vector Machine (SVM), Gaussian Process Regression (GPR), etc (Rahimi & Recht, 2007). In this paper, we conduct experiments to investigate the effect of dimensionality and sampling distribution on the performance of RFF. We also compare the performance of RFF with other kernel methods, such as Support Vector Machine (SVM), Gaussian Process Regression (GPR), etc. We find that RFF is sensitive to the dimensionality and sampling distribution. We also find that RFF is comparable to other kernel methods in terms of accuracy, but it is much faster than other kernel methods in terms of training and inference time.

The rest of this paper is organized as follows. In Section 2, we review related works. In Section 3, we introduce the preliminaries of RFF. In Section 4, we describe the experiments we conducted. In Section 5, we present the results of our experiments. In Section 6, we conclude our paper.

*Equal Contribution. Wang Ma contributed the main ideas and led others, Qiang Hu was responsible for the random feature experiment and analysis, Jingjuan Huang was responsible for the GPR experiment and analysis, and Junjie Qiu assisted Hu Qiang in the experiment and led the writing.

Our intuitions and motivations of following experiments are mainly inspired Jin et al. (2022) and Li et al. (2022), thanks for their pioneering works!

Our main contributions are as follows:

- We conduct experiments to investigate the effect of dimensionality and sampling distribution on the performance of RFF.
- We compare the performance of RFF with other kernel methods, such as Support Vector Machine (SVM), Gaussian Process Regression (GPR), etc.
- We find that RFF is sensitive to the dimensionality and sampling distribution.
- We find that RFF is comparable to other kernel methods in terms of accuracy, but it is much faster than other kernel methods in terms of training and inference time.
- We find that RFF models are comparable to neural network in $R^2$, SHAP value and Permutation Feature Importances.

## 2 Related Works

**Random Fourier Features.** Rahimi & Recht (2007) proposed Random Fourier Features (RFF) to approximate the kernel function $k(\cdot, \cdot)$ by a random feature map $\phi(\cdot)$. It is considered as a breakthrough in kernel methods, since it makes kernel methods scalable to large datasets. It is widely used in kernel methods, such as Support Vector Machine (SVM), Gaussian Process Regression (GPR), etc (Rahimi & Recht, 2007).

**Dimensions and Sampling Distributions:** The dimensions of features and properties of sampling distributions play a crucial role in the performance of RFF-based models. Studies have shown that larger dimensions can lead to increased accuracy but at the cost of higher computational requirements (Gundersen (2019); Bottou (2018)).

**RFF in Advanced Models:** The application of RFF in advanced models like Efficient Transformers showcases the versatility and potential of RFF in enhancing model performance and efficiency. Recent works like performer (Choromanski et al., 2020) also use random features to approximate the attention mechanism in Transformer, showing the effectiveness of this approach Koker (2020).

## 3 Preliminaries

**Random Fourier Features** (Rahimi & Recht, 2007). Random Fourier Features (RFF) is a method to approximate the kernel function $k(\cdot, \cdot)$ by a random feature map $\phi(\cdot)$, which is defined as

$$\varphi(x) = \sqrt{\frac{2}{D}}[\cos(\eta_1^\top x + b_1), \cos(\eta_2^\top x + b_2), ..., \cos(\eta_D^\top x + b_D)], \ \eta \sim p(\eta) \tag{1}$$

where $\eta \in \mathbb{R}$ is a random vector and $b \in \mathbb{R}$ is a random scalar Avron et al. (2017).

**Radial Basis Function (RBF) kernel functions.** In the domain of kernel methodsGundersen (2019), the Radial Basis Function (RBF) kernel stands out due to its remarkable properties in capturing the non-linear relationships in data. The RBF kernel, also known as the Gaussian kernel, is defined as $K(x, x') = \exp(-\gamma\|x - x'\|^2)$, where $\gamma$ is a scale parameter and $x, x'$ are data points. This kernel function, owing to its infinite dimensionality, facilitates an effective mapping of data into a higher-dimensional spaceBottou (2018), enabling the linear separation of non-linearly separable data patterns Buhmann (2003).

**Remark on Coefficient of Determination.** The coefficient of determination, denoted as $R^2$, measures the proportion of the variance in the dependent variable that is predictable from the independent variables in a regression model. It is defined as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2}$$

where $n$ is the number of observations, $y_i$ is the observed value of the dependent variable for observation $i$, $\hat{y}_i$ is the predicted value of the dependent variable for observation $i$, and $\bar{y}$ is the mean of the observed values of the dependent variable.

It is important to note that when estimating $R^2$ without utilizing the method of Least Squares Estimation (LSE), the coefficient of determination may **assume negative values**. This can occur due to the absence of a systematic relationship between the independent and dependent variables, leading to a model that performs worse than a simple mean-based predictor.

**SHAP Values.** SHAP values is to quantify the impact of each feature by considering its contribution to every possible prediction. This holistic approach ensures that the contributions are fairly distributed among the features, providing a clear understanding of the importance of each variable in the model's decision-making process. Mathematically, for a prediction $f(x)$ made by a model with features $x$, SHAP values are defined as the average marginal contribution of a feature across all possible feature combinations:

$$\text{SHAP}_i(f) = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \left[ f(S \cup \{i\}) - f(S) \right] \tag{3}$$

where $N$ is the set of all features, $S$ is a subset of features excluding $i$, and $f(S)$ represents the model's prediction when considering only the features in subset $S$.

**Permutation Feature Importance** (Altmann et al., 2010). Permutation Feature Importance (PFI) is a widely used technique in machine learning to assess the importance of individual features in predictive models. Developed as a robust and model-agnostic approach, PFI provides insights into the influence of each feature on the model's performance by evaluating the impact of feature permutations on predictive accuracy. The fundamental idea behind PFI involves systematically permuting the values of a single feature while keeping the others unchanged and measuring the resulting change in model performance. Specifically, for a model $f(\mathbf{x})$ where $\mathbf{x}$ represents the feature vector, the Permutation Feature Importance $I_i$ for the $i$-th feature is computed as:

$$I_i = \frac{1}{K} \sum_{k=1}^{K} (\text{score}_k - \text{score}_{k,\text{permuted}}) \tag{4}$$

where $\text{score}_k$ is the model's performance metric (e.g., accuracy) on the original dataset, and $\text{score}_{k,\text{permuted}}$ is the performance metric on the dataset with the $i$-th feature permuted. $K$ denotes the number of permutations, and the larger the sample size, the more accurate the estimation of feature importance.

## 4 Experiments

### 4.1 Experiments Setup

**Datasets.** For classification, we consider MNIST (LeCun & Cortes, 2010). The MNIST dataset is a widely used collection of handwritten digit images. MNIST contains a set of 28x28 pixel grayscale images of handwritten digits (0 through 9). The dataset is split into a training set with 60,000 examples and a test set with 10,000 examples. Each image is labeled with the corresponding digit it represents.

For regression, we consider dataset of California Housing. It provides a comprehensive set of features(8 features including Hosing Price) related to housing characteristics in various districts across California.

**Models.** We employ Support Vector Classifier(svc) (Cortes & Vapnik, 1995), Gaussian Process Regression(GPR) (Williams & Rasmussen, 1995), RFFRegression (Rahimi & Recht, 2007) and a customed neural network. And for svc, we consider the kernels including linear, radial basis function, polynomial and sigmoid.

**Evaluation.** The existing metrics used to evaluate the performance of a kernel machine includes training time, inference time, accuracy score and $R^2$, where training time and inference time indicates the time spent to fit the model and classify a single example respectively. For classification the accuracy score is applied and it can be formally represented as:

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}}$$

For Regression, $R^2$ is applied and it can be represented as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

### 4.2 RESULTS

***Performance of 4 baseline kernel methods for SVC***

| Kernel | linear | rbf | polynomial | sigmoid |
|---|---|---|---|---|
| Training time [sec] | 33.1 | **24.3** | 18.7 | 64.6 |
| Inference time [us] | **306.6** | 1102.6 | 340.1 | 1353.5 |
| Score [%] | 94.5 | 97.4 | **97.9** | 84.93 |

Table 1: Performance of baseline kernel methods for SVC on mnist.

Among the four baseline kernel considered here, $polynomial \simeq rbf > linear > sigmoid$ in terms of accuracy score, and $sigmoid > rbf >> polynomial > linear$ in terms of inference time. We noticed that the inference consumed a lot for all four methods, and that's where rff will improve significantly.

#### 4.2.1 INVESTIGATING TO THE DIMENSIONS OF FEATURES

We conduct an experiment to explore the effect of kernel dimension on Performance of rff. Here the gaussian sampling and uniform sampling methods are chosen, and kernel dimension varies from 80 to 4096.

| Kernel Dimension | 80 | 160 | 320 | 640 | 1024 | 4096 |
|---|---|---|---|---|---|---|
| Training time[sec] | **7.3** | 14.7 | 20.5 | 31.6 | 41.6 | 126.6 |
| Inference time[us] | **7.7** | 10.2 | 19.4 | 23.0 | 30.6 | 109.6 |
| Score[%] | 89.0 | 92.7 | 95.0 | 96.5 | 97.2 | **98.3** |

Table 2: Performance of rff via Gaussian sampling methods for SVC on mnist

It can be seen that larger kernel dimension leads to higher score, as well as more train and inference time. This makes sense because higher dimensions can represent more detail, leading to higher accuracy. Unless additional statements, we use 1024 dimensions for the following experiments because of its high accuracy and acceptable training and reasoning time.

| Kernel dimension | 80 | 160 | 320 | 640 | 1024 | 4096 |
|---|---|---|---|---|---|---|
| Training time [sec] | **9.7** | 16.2 | 31.1 | 70.8 | 130.0 | 335.7 |
| Inference time [us] | **7.0** | 12.6 | 10.2 | 28.0 | 27.3 | 104.6 |
| Score [%] | 88.2 | 92.2 | 93.4 | 94.6 | 95.1 | **96.0** |

Table 3: Performance of rff via uniform sampling methods for SVC on mnist

For uniform sampling, it has a similar trend to Gaussian sampling. However, under the same dimension, it takes longer training time and has a consistently lower accuracy score than Gaussian sampling.

#### 4.2.2 Investigating to the Properties of Sampling Distributions

We change the standard deviation and mean of the distribution in Gaussian sampling to explore the effect of the distribution properties on the performance of rff. It can be formulated as follows:

$$W = \sigma \cdot A + \mu$$

where W is the rff matrix, $A \sim \mathcal{N}_{m \times n}(0, 1)$, m and n are the input and output dimensions of the rff matrix, $\sigma$ and $\mu$ are the standard deviation and mean of Gaussian sampling respectively.

***Influence of Std of Sampling Distribution***

Here we alter $std$ from 0 to 5, and records training time, inference time and accuracy score for each std value. The following are the visualization of the result, and the exact numerical results are in the appendix.

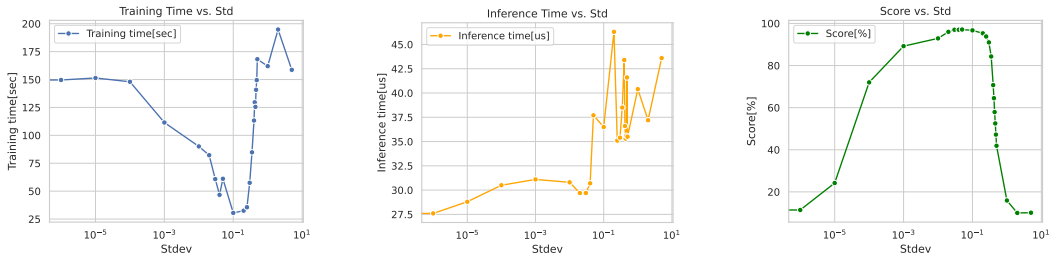



Figure 1: Performance of rff via Gaussian sampling for SVC on MNIST(std $0 \sim 5$)

- For training time, a trough occurs when $std \simeq 0.1$. And both lower or higher std value would leads to significantly larger time consumption.
- The inference time remain a stable level until $std \simeq 0.05$, after which it increased rapidly and was accompanied by large fluctuations.
- The score is stably on high level between 0.001 and 0.1, and when the std is too large or too small it can cause the score to drop rapidly to a random guess level.

Considering these three metrics(training time, inference time and accuracy score) comprehensively, we could conclude that rbf with $Std \simeq 0.05$ perform best.

***Influence of Mean of Sampling Distribution***

We experiment the influence of mean in Gaussion sampling. Here we change the mean from 0 to 0.15 for $kdim = 1024$, and 0 to 0.075 for $kdim = 128$ respectively.

| Mean | 0 | 0.025 | 0.05 | 0.075 | 0.1 | 0.125 | 0.15 |
|---|---|---|---|---|---|---|---|
| Training time [sec] | 46.0 | **42.1** | 43.6 | 48.4 | 48.7 | 55.74 | 67.2 |
| Inference time [us] | 33.2 | 31.0 | **29.9** | 31.1 | 30.6 | 36.6 | 33.1 |
| Score [%] | **97.25** | 97.1 | 97.0 | 96.6 | 96.2 | 95.3 | 93.2 |

Table 4: Performance of rbf-rff with different mean on mnist. The kernel dimension is 1024.

We assert that the overall effect is best when mean is near 0.05, when the three trade-off metrics reach balance, with smallest inference time and relatively good training time(+1.5sec) and score(-0.25%).

| Mean | 0 | 0.025 | 0.05 | 0.075 |
|---|---|---|---|---|
| Training time [sec] | **11.0** | 11.6 | 11.5 | 13.0 |
| Inference time [us] | 6.8 | 6.7 | **5.8** | 7.8 |
| Score [%] | 91.5 | 91.4 | 90.9 | **90.3** |

Table 5: Performance of rbf-rff with mean on mnist. The kernel dimension is 128.

The outcome for $kdim = 128$ is similar to that when $kdim = 1024$, and $mean \simeq 0.05$ also leads to the best performance, which indicates that our results are unrelated to kernel dimensions within a certain range.

#### 4.2.3 COMBINATION OF HIGH FREQUENCY AND LOW FREQUENCY

As mentioned before, we assume that low-frequency information tends to describe the outline of the data set, while high-frequency information tends to describe the details of the data set. Since that, we try to combine high-frequency and low-frequency features, and study the effect of rff under such integrated features. It can be formulated as follows:

$$W = \sigma \cdot [A, B]$$

where $A \sim \mathcal{N}_{m\times 0.5n}(0, 1)$ and $B \sim \mathcal{N}_{m\times 0.5n}(bias, 1)$, m and n are dimension of input and output of rff matrix $W$ respectively. And here we change bias from 0.01 to 10.

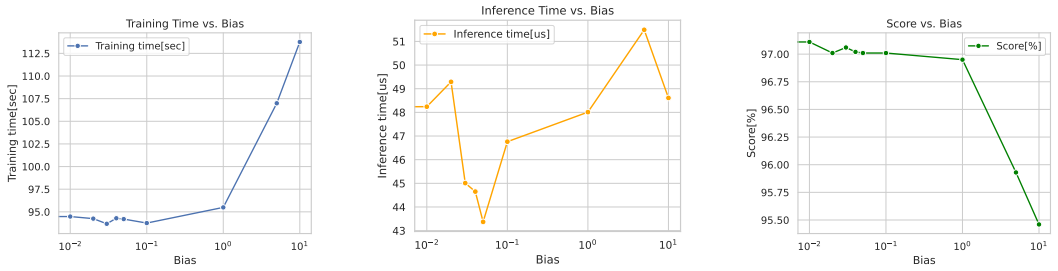



Figure 2: Visualization for performance of rff with the combination of high frequency and low frequency, via Gaussian sampling for SVC on MNIST, kernel dim 1024

- The training time increases with the increase of bias. When bias$<$1, training time maintained at a low level; When bias is greater than 1, the training time increases rapidly (from about 95s to 113.76s).
- When bias varies from 0.01 to 10, the inference time drops first and then rises, reaching the minimum when $bias \simeq 0.05$
- When the bias is less than 1, the accuracy score remains almost unchanged. However, when the bias is greater than 1, the accuracy score begins to decline significantly.

The exact numerical results are in the following table. Therefore, we assert that $bias \simeq 0.05$ is the best choice, since the inference time decreased a lot(-4.87s), while the training time and score almost keep unchanged($< 1\%$) compared with $bias \simeq 0$

| Bias | Training time[sec] | Inference time[us] | Score[%] |
|---|---|---|---|
| 0 | **93.61** | 45.14 | 97.08 |
| 0.01 | 94.48 | 48.24 | **97.11** |
| 0.02 | 94.25 | 49.29 | 97.01 |
| 0.03 | 93.68 | 45.01 | 96.06 |
| 0.04 | 94.29 | 47.55 | 97.03 |
| 0.05 | 94.20 | **43.37** | 97.01 |
| 0.1 | 93.76 | 46.76 | 97.01 |
| 1 | 95.49 | 48.01 | 96.95 |
| 5 | 106.99 | 51.49 | 95.93 |
| 10 | 113.76 | 48.61 | 95.46 |

Table 6: Performance of rff via Gaussian sampling for SVC on MNIST, kernel dimension 1024

#### 4.2.4 Analysis of California Housing Price

***Impact of Dimensionality on RFFGPR and RFFRegression $R^2$ Scores***

We employed the RFFGPR and RFFRegression methods to manipulate the kernel dimension. The observed outcomes of the fitting performance are as follows:

- At lower dimensions, RFFRegression outperforms.
- At higher dimensions, RFFGPR exhibits superior performance.
- With an increase in dimensionality, the performance of RFFGPR consistently improves, while RFFRegression initially shows enhancement followed by a decline.
- It is interesting to see that RFFGPR maintains improvement even at dimensionality 1024; in contrast, RFFRegression experiences a decline in $R^2$ starting from dimensionality 128.

| Kernel dimension | 8 | 16 | 32 | 64 | 128 | 256 | 640 | 1024 |
|---|---|---|---|---|---|---|---|---|
| RFFGPR | 0.5498 | 0.6258 | 0.65 | 0.6604 | 0.6814 | 0.694 | 0.6972 | **0.6986** |
| RFFRegression | 0.5529 | 0.6258 | 0.647 | **0.6685** | 0.6611 | 0.6289 | -3.2463 | -67.5435 |

Table 7: $R^2$ of RFFGPR & RFFRegression in different dimension

***Variation of $R^2$ in Neural Network Across Epochs***

We customized a neural network and observed that: As the number of epochs increases, there is a continuous improvement in the $R^2$ score. The increasing trend in $R^2$ underscores the efficacy of the neural network in capturing underlying patterns and enhancing its predictive capabilities over successive epochs.

| Epochs | 1 | 10 | 50 | 100 | 200 | 500 | 1000 | 2000 | 5000 |
|---|---|---|---|---|---|---|---|---|---|
| Values | -2.7107 | -2.3972 | -0.0652 | 0.394 | 0.5639 | 0.6782 | 0.7111 | 0.7374 | **0.7711** |

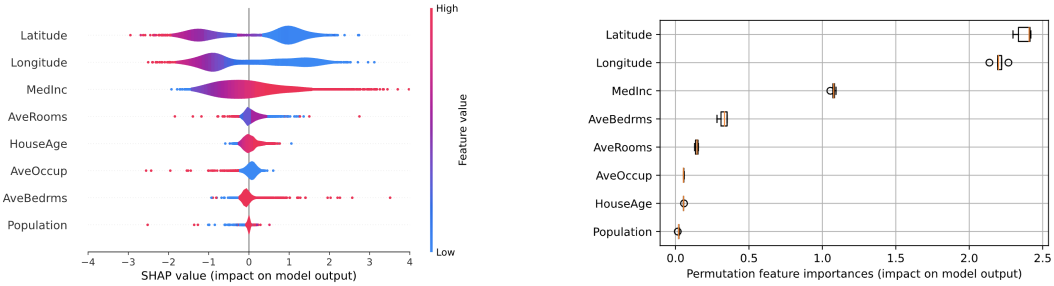
Table 8: $R^2$ of NN in different epochs

***Model Comparison***

We compared models with similar $R^2$ scores, specifically employing RFFGPR with a dimensionality of 128, RFFRegression with a dimensionality of 64, and a neural network trained for 500 epochs. The analysis, utilizing SHAP values and Permutation Feature Importances, revealed the following observations:

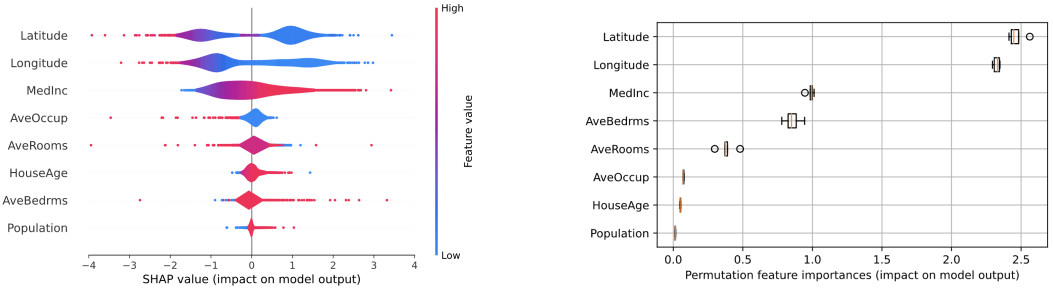
- Utilizing SHAP Values for Feature Extraction: Both models identified Latitude, Longitude, and Medinc as the top three most important features. Both models ranked AveBedrms and Population as the least important features.

- Utilizing Permutation Feature Importances: The importance rankings of features extracted by the three models are identical.



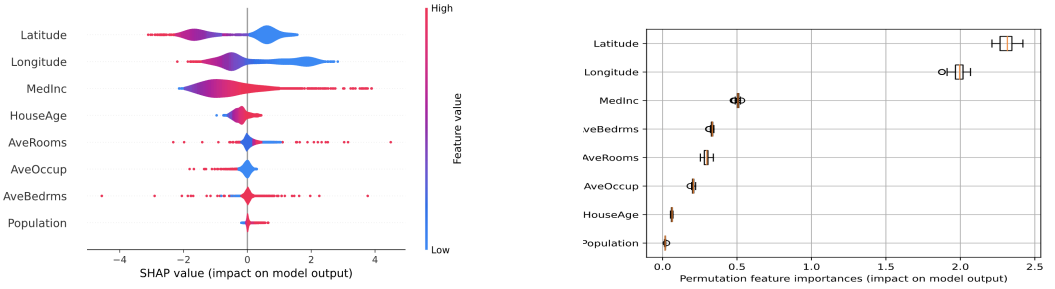

(a) SHAP Values (b) Permutation Importance

Figure 3: RFFGPR (dim = 128)




(a) SHAP Values (b) Permutation Importance

Figure 4: RFFRegression (dim = 64)




(a) SHAP Values (b) Permutation Importance

Figure 5: Neural Network (500 epochs)

## 5 Conclusions

In this paper, we investigate the effect of dimensionality and sampling distribution on the performance of RFF. We also compare the performance of RFF with other kernel methods, such as Support Vector Machine (SVM), Gaussian Process Regression (GPR), etc. We find that the choices and properties of sampling distributions impacts the performance of RFF mainly in the selection of features. After conducting experiments evaluating feature significance within RFF framework, we find that the feature selection capability of RFF is comparable to well-trained Neural Networks on California Housing Prices dataset. Our work provides an insight into the impact of random feature selection and stimulates further research into the improvement of RFF.

## REFERENCES

André Altmann, Laura Toloşi, Oliver Sander, and Thomas Lengauer. Permutation importance: a corrected feature importance measure. *Bioinformatics*, 26(10):1340–1347, 2010.

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*, pp. 253–262. PMLR, 2017.

Léon Bottou. The radial basis function kernel, 2018. URL https://pages.cs.wisc.edu/~matthewb/pages/notes/pdf/svms/RBFKernel.pdf. Accessed: 2024-01-05.

M. D. Buhmann. *Radial Basis Functions: Theory and Implementations*. Cambridge University Press, 2003.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers, 2020.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

Gregory Gundersen. Implicit lifting and the kernel trick, 2019. URL https://gregorygundersen.com/blog/2019/12/10/kernel-trick/. Accessed: 2024-01-05.

Jikai Jin, Yiping Lu, Jose Blanchet, and Lexing Ying. Minimax optimal kernel operator learning via multilevel training. *arXiv preprint arXiv:2209.14430*, 2022.

Teddy Koker. Performers: Rethinking attention. *Blog post*, 2020. URL https://teddykoker.com/2020/11/performers/. Accessed: 2024-01-05.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database. 2010. URL http://yann.lecun.com/exdb/mnist/.

Kexin Li, Hongwei Li, Raymond H. Chan, and Youwei Wen. Selecting regularization parameters for nuclear norm–type minimization problems. *SIAM Journal on Scientific Computing*, 44(4): A2204–A2225, 2022. doi: 10.1137/21M143786X. URL https://doi.org/10.1137/21M143786X.

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

Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995.

# A APPENDIX

| Std | Training time[sec] | Inference time[us] | Score[%] |
|---|---|---|---|
| 0.05 | 61.1 | 37.7 | 97.1 |
| 0.1 | 30.5 | 36.5 | 96.7 |
| 0.2 | 32.5 | 46.3 | 95.3 |
| 0.25 | 35.6 | 35.1 | 93.8 |
| 0.3 | 57.5 | 35.4 | 91.0 |
| 0.35 | 84.8 | 38.5 | 84.3 |
| 0.4 | 113.3 | 43.4 | 70.7 |
| 0.42 | 129.6 | 36.6 | 64.5 |
| 0.44 | 125.5 | 35.3 | 57.9 |
| 0.46 | 140.8 | 35.6 | 52.5 |
| 0.48 | 149.5 | 41.6 | 47.2 |
| 0.5 | 168.2 | 35.5 | 41.9 |
| 1 | 162.0 | 40.4 | 15.9 |
| 2 | 194.8 | 37.2 | 10.0 |
| 5 | 158.7 | 43.6 | 10.1 |

Table 9: Performance of rff via gaussian methods for SVC on mnist (Transposed)