Error Estimation for Random Fourier Features

Junwen Yao
University of California, Davis

N. Benjamin Erichson
Lawrence Berkeley National Laboratory

Miles E. Lopes
University of California, Davis

Abstract

Random Fourier Features (RFF) is among the most popular and broadly applicable approaches for scaling up kernel methods. In essence, RFF allows the user to avoid costly computations on a large kernel matrix via a fast randomized approximation. However, a pervasive difficulty in applying RFF is that the user *does not know the actual error* of the approximation, or how this error will propagate into downstream learning tasks. Up to now, the RFF literature has primarily dealt with these uncertainties using theoretical error bounds, but from a user's standpoint, such results are typically impractical—either because they are highly conservative or involve unknown quantities. To tackle these general issues in a data-driven way, this paper develops a bootstrap approach to *numerically estimate* the errors of RFF approximations. Three key advantages of this approach are: (1) The error estimates are specific to the problem at hand, avoiding the pessimism of worst-case bounds. (2) The approach is flexible with respect to different uses of RFF, and can even estimate errors in downstream learning tasks. (3) The approach enables adaptive computation, so that the user can quickly inspect the error of a rough initial kernel approximation and then predict how much extra work is needed. Lastly, in exchange for all of these benefits, the error estimates can be obtained at a modest computational cost.

1 INTRODUCTION

Although kernel methods are fundamental to many types of machine learning systems, they have an Achilles heel, insofar as they have limited scalability when they are applied to large datasets in a direct manner (Schölkopf and Smola, 2002; Shawe-Taylor and Cristianini, 2004). The basic source of this issue is that an $n \times n$ kernel matrix derived from n data points typically incurs an $\mathcal{O}(n^2)$ storage cost, and an $\mathcal{O}(n^3)$ processing cost for common learning tasks. Due to such bottlenecks, techniques for accelerating kernel methods have been studied extensively, and over the years, the approach of *Random Fourier Features* (RFF) has become well-established as one of the most popular and effective ways to scale up kernel methods in a plethora of applications (Rahimi and Recht, 2007; Le et al., 2013; Dai et al., 2014; Zhao and Meng, 2015; Avron et al., 2017; Zhang et al., 2019; Liu et al., 2021; Giannakis et al., 2022; Kiessling et al., 2021).

The core idea of RFF is to avoid direct computations on a large kernel matrix by working more efficiently with an approximation built from "randomly sampled features". As a result of this approximation, RFF involves an inherent tradeoff between computational cost and accuracy. However, managing this tradeoff in practice is complicated by the fact that the user *does not know the actual error* of the approximation, or how this error may jeopardize downstream results. In addition, this uncertainty about error can lead the user to sample far more features than are really necessary, which erodes the computational gains of RFF.

At a conceptual level, the RFF literature is able to offer insights on these issues with various theoretical error bounds, which are surveyed in Liu et al. (2021). However, there has been a longstanding gap between theory and practice, because these results generally do not provide actionable guidance at a numerical level. One reason for this difficulty is that theoretical error bounds tend to be formulated to hold uniformly over a class of possible inputs, which often causes the bounds to be highly pessimistic for typical problem instances. (Empirical illustrations of this conservativeness can be found, for example, in Figures 4 and 5 of Sutherland and Schneider (2015).) Such bounds frequently also involve unspecified constants or unknown parameters, preventing the user from extracting any numerical information at all.

Contributions. To overcome the challenges above, we develop a systematic way to *numerically estimate* the errors of RFF approximations. Our contributions are briefly summarized below.

Proceedings of the 26th International Conference on Artificial Intelligence and Statistics (AISTATS) 2023, Valencia, Spain. PMLR: Volume 206.

1. The error estimates are fully-data driven, and hence tailored to the inputs in a given problem. This bypasses practical limitations of worst-case error bounds.
2. The error estimates enhance the computational efficiency of RFF, by guiding the user to choose a number of features that is just enough for a preferred error tolerance.
3. We give a precise theoretical guarantee on the validity of the error estimates in the context of kernel matrix approximation (Theorem 1), holding under mild assumptions.
4. We demonstrate the versatility of the error estimates in several RFF use cases, including kernel matrix approximation, kernel ridge regression, and kernel-based hypothesis testing.

1.1 Peliminaries on kernels and RFF

Kernels. Throughout the paper, we consider learning tasks involving a shift-invariant kernel $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$. This means that k is positive definite and satisfies the relation $k(x, x') = k(x - x', 0)$ for all $x, x' \in \mathbb{R}^d$. In addition, we will always assume that k is continuous and is normalized so that $k(0, 0) = 1$. Kernels with these properties are the ones most often studied in the RFF literature, and well-known examples include the Gaussian, Laplacian, Cauchy, and B-spline kernels, among others surveyed in §4.4-4.5 of Schölkopf and Smola (2002).

Random features. From a mathematical perspective, the linchpin of RFF is a classical result from Fourier analysis known as Bochner's Theorem, which ensures that if k is of the stated type, then there exists a probability distribution ρ on $\mathbb{R}^d$ such that k can be represented as

$$k(x, x') \;=\; \int_{\mathbb{R}^d} e^{\sqrt{-1}\langle x-x', w\rangle} \mathrm{d}\rho(w), \tag{1.1}$$

for all $x, x' \in \mathbb{R}^d$, where $\langle \cdot, \cdot \rangle$ is the Euclidean inner product (Rudin, 1990). Crucially, this integral representation allows the kernel to be viewed as an expectation, because if W is a random vector drawn from ρ, and if we define the "random feature" $\zeta(x) = e^{\sqrt{-1}\langle x, W\rangle}$ for any fixed $x \in \mathbb{R}^d$, then $k(x, x') = \mathbf{E}[\zeta(x)\zeta(-x')]$. However, due to the fact that k is real-valued, whereas $\zeta(x)$ is complex-valued, it has become common in the RFF literature to use a real-valued modification of $\zeta(x)$. Such a modification can be defined as

$$Z(x) = \sqrt{2}\cos(\langle x, W\rangle + U),$$

where U is drawn from the uniform distribution on $[0, 2\pi]$ independently of W, leading to

$$k(x, x') \;=\; \mathbf{E}[Z(x)Z(x')]. \tag{1.2}$$

Randomized kernel approximations. The importance of viewing k as an expectation is that it enables us to approximate k with a sample average involving s random features, where $s \ll n$. Specifically, let $W_1, \ldots, W_s \sim \rho$ and $U_1, \ldots, U_s \sim \text{Uniform}[0, 2\pi]$ be independent sets of i.i.d. samples, and for any fixed $x \in \mathbb{R}^d$, denote the associated random features as $Z_i(x) = \sqrt{2}\cos(\langle x, W_i\rangle + U_i)$, with $i = 1, \ldots, s$. In this notation, the RFF approximation to $k(x, x')$ is defined by

$$\tilde{k}(x, x') \;=\; \frac{1}{s}\sum_{i=1}^{s} Z_i(x)Z_i(x'), \tag{1.3}$$

which is unbiased, $\mathbf{E}[\tilde{k}(x, x')] = k(x, x')$, due to (1.2).

Regarding kernel matrices, consider a fixed set of data points $x_1, \ldots, x_n \in \mathbb{R}^d$, and let $\mathsf{K} \in \mathbb{R}^{n\times n}$ have entries given by $\mathsf{K}_{jj'} = k(x_j, x_{j'})$. An approximation to K can be developed by first defining a random matrix $\mathsf{Z} \in \mathbb{R}^{n\times s}$ whose ith column is $\frac{1}{\sqrt{s}}(Z_i(x_1), \ldots, Z_i(x_n))$. Then, in light of (1.3), the RFF approximate kernel matrix is defined as

$$\widetilde{\mathsf{K}} \;=\; \mathsf{Z}\mathsf{Z}^\top. \tag{1.4}$$

To briefly describe the computational advantages of RFF, it is worth re-emphasizing that the number of random features s is generally chosen so that $s \ll n$. Combining this with the fact that $\widetilde{\mathsf{K}}$ is automatically factorized in terms of the $n \times s$ matrix Z, it follows that for any $v \in \mathbb{R}^n$, a matrix-vector product can be computed as $\widetilde{\mathsf{K}}v = \mathsf{Z}[\mathsf{Z}^\top v]$ with a cost of only $\mathcal{O}(sn)$. Hence, this is much less than the corresponding $\mathcal{O}(n^2)$ cost to compute $\mathsf{K}v$. More generally, such savings in linear-algebraic operations have enabled RFF to speed up a variety of learning tasks—such as reducing cost from $\mathcal{O}(n^3)$ to $\mathcal{O}(s^2 n)$ in both kernel PCA and kernel ridge regression (Lopez-Paz et al., 2014; Avron et al., 2017).

1.2 Formalizing the error estimation problem

Errors with respect to norms. When assessing the error of $\widetilde{\mathsf{K}}$ in relation to the exact matrix K, a variety of norms may be of interest. Since our approach is flexible with respect to this choice, we let $\|\cdot\|_\diamond$ denote a generic norm on $\mathbb{R}^{n\times n}$. For any such choice, it should be stressed that the actual error $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\diamond$ is both *random* and *unknown* to the user. Also, we regard the exact matrix K as fixed, and so the randomness in $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\diamond$ arises entirely from the random features used to construct $\widetilde{\mathsf{K}}$.

Our goal is to numerically estimate the tightest possible upper bound on $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\diamond$ that holds with a given probability, say $1 - \alpha$, where $\alpha \in (0, 1)$. More formally, this ideal (unknown) bound is called the $(1-\alpha)$-quantile of $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\diamond$, and is defined as

$$\varepsilon_{1-\alpha} \;=\; \inf\Big\{c \in [0, \infty) \;\Big|\; \mathbf{P}\Big(\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\diamond \le c\Big) \;\ge\; 1 - \alpha\Big\}.$$

Below, Figure 1 illustrates how the quantile $\varepsilon_{1-\alpha}$ can be interpreted in relation to the fluctuations of the random

variable $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\diamond$, in the particular case when $\|\cdot\|_\diamond$ is the operator (spectral) norm and $1-\alpha = 90\%$.

To explain Figure 1, consider a hypothetical scenario where it is possible to track the random variable $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\diamond$ as the number of features s is increased over a grid ranging from 1 to 1600. The result is displayed with the red curve. Similarly, by repeating this experiment many times, a large collection of such random curves can be generated, and these are displayed in blue. (This scenario would not occur in practice, and is only for conceptual illustration.) In addition, the 90% quantile of the curves at each value of s is plotted in black, which represents $\varepsilon_{1-\alpha}$.

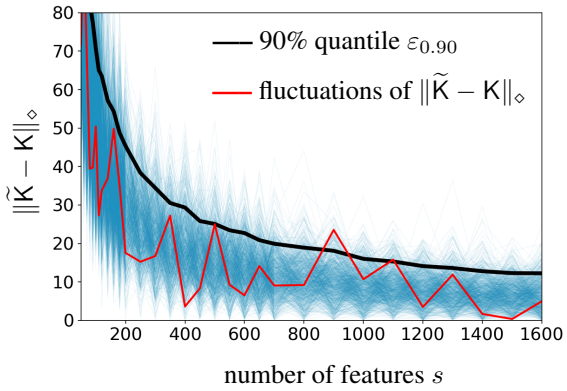

Figure 1: (Interpretation of $\varepsilon_{1-\alpha}$ when $1-\alpha = 90\%$.) The plot was generated in the setting of the "Swiss roll" dataset described in Appendix C. In particular, this involved a Gaussian kernel with bandwidth $\sigma = 4\%$.

Hence, if the user had access to the black curve for $\varepsilon_{1-\alpha}$, it would be possible to know if a given number of features s_0 is adequate, or to predict what larger value $s > s_0$ should be used to achieve a higher level of accuracy. Despite the fact that none of the curves in Figure 1 are available to the user in practice, our work will show that, for a given value s_0, there is enough information in a *single instance* of the $n \times s_0$ matrix Z (as in (1.4)) to closely estimate $\varepsilon_{1-\alpha}$ for that value of s_0. Furthermore, it will also be shown in Section 3.1 that a simple extrapolation rule can be used to rapidly estimate $\varepsilon_{1-\alpha}$ for all larger values $s > s_0$.

Estimation criteria. When computing a numerical estimate, say $\tilde{\varepsilon}_{1-\alpha}$, for the true quantile $\varepsilon_{1-\alpha}$, there are several important criteria to be met. First, the estimate should serve as a good proxy for $\varepsilon_{1-\alpha}$, in the sense that the inequality

$$\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\diamond \;\leq\; \tilde{\varepsilon}_{1-\alpha}$$

holds with a probability that is close to $1-\alpha$ (cf. Theorem 1). Second, the estimate should not require any access to the full kernel matrix K. Third, the algorithm used to compute $\tilde{\varepsilon}_{1-\alpha}$ should be efficient, so that the cost of error estimation does not outweigh the benefit of using RFF. In the remainder of this work, our proposed approach will be shown to meet all of these criteria.

Errors with respect to functionals. In addition to measuring error through norms, it is also of interest to measure error by comparing the kernel functions $\tilde{k}$ and k with respect to various functionals ψ. For example, the values $\psi(\tilde{k})$ and $\psi(k)$ could be measures of prediction error for learning algorithms based on $\tilde{k}$ and k respectively, so that the difference $\psi(\tilde{k}) - \psi(k)$ represents how much predictive performance is sacrificed by the RFF approximation. More generally, there are many other possibilities for comparing $\tilde{k}$ and k in different contexts, such as letting ψ represent eigenvalues in kernel PCA, or letting ψ represent statistics for testing hypotheses. In these broader scenarios, the previous formulation of the error estimation problem can be extended by defining a counterpart for $\varepsilon_{1-\alpha}$ according to

$$\delta_{1-\alpha} = \inf\Big\{c \in [0,\infty) \,\Big|\, \mathbf{P}\Big(|\psi(\tilde{k}) - \psi(k)| \leq c\Big) \geq 1-\alpha\Big\}.$$

Likewise, our proposed approach can be applied to compute an estimate $\tilde{\delta}_{1-\alpha}$ for $\delta_{1-\alpha}$ that meets the criteria mentioned previously. Also, the approach can be applied just as easily if the user prefers to define $\delta_{1-\alpha}$ by replacing $|\psi(\tilde{k}) - \psi(k)|$ with $\psi(\tilde{k}) - \psi(k)$.

1.3 Related work

The existing literature on theoretical error bounds for RFF has grown substantially over the years, and so we only provide an illustrative sample. Results on kernel approximation can be found in Rahimi and Recht (2007); Sutherland and Schneider (2015); Sriperumbudur and Szabó (2015); Liu et al. (2021). With regard to error analysis in other applications, such as such as kernel-based regression, classification, and hypothesis testing, we refer to Yang et al. (2012); Sutherland and Schneider (2015); Avron et al. (2017); Rudi and Rosasco (2017); Sun et al. (2018); Li et al. (2019); Liu et al. (2021).

To situate the current paper in the general context of numerical computation, our work can be viewed as part of a topic known as *a posteriori error estimation*—which refers to the process of estimating the error of a numerical solution after it has been computed. Although this topic has a mature literature in areas such as numerical PDE and finite element methods, an important distinction to make is that a posteriori error estimation has focused historically on deterministic algorithms (e.g. Babuška and Rheinboldt, 1978; Bank and Weiser, 1985; Verfürth, 1994; Ainsworth and Oden, 2011). Meanwhile, from a different perspective, our work can also be viewed as part of the extensive literature on *bootstrap methods* for statistical inference (e.g. Davison and Hinkley, 1997; Hall, 2013; Shao and Tu, 2012). Yet, from the standpoint of the statistics literature, relatively little attention has

been given to bootstrap methods in the service of randomized algorithms for large-scale computation. Hence, our work sits at the border of two fields that have traditionally been quite distinct.

Nevertheless, the possibility of bridging this gap has not been overlooked completely, and there has been nascent interest in applying statistical ideas to estimate the errors of randomized algorithms, as noted in the recent survey (Martinsson and Tropp, 2020). For instance, such interest has led to error estimation methods for randomized solutions to low-rank approximation (Liberty et al., 2007; Woolfe et al., 2008; Halko et al., 2011), matrix multiplication (Lopes et al., 2019, 2023), least-squares (Lopes et al., 2018; Ahfock et al., 2021), singular value decomposition (Lopes et al., 2020), and principal component analysis (Lunde et al., 2021). However, to the best of our knowledge, statistical error estimation techniques for RFF have not previously been explored in a systematic way. Therefore, given that RFF has been highly impactful, our work may offer new opportuntities to enhance many applications.

Notation. For any $\alpha \in (0,1)$, the empirical $(1-\alpha)$-quantile of a finite set of real numbers $A = \{a_1, \dots, a_N\}$ is defined as the smallest $a \in A$ such that $G_N(a) \geq 1-\alpha$, where $G_N(a) = \frac{1}{N}\sum_{j=1}^{N} 1\{a_j \leq a\}$, and $1\{\cdot\}$ denotes an indicator function. To denote this quantile, we write quantile$(A; 1-\alpha)$. If $\mathbf{i} = (i_1, \dots, i_s)$ is a vector with entries taken from $\{1, \dots, s\}$, then $\mathsf{Z}(:, \mathbf{i})$ refers to the $n \times s$ matrix whose lth column is the i_lth column of $\mathsf{Z} \in \mathbb{R}^{n \times s}$. Similarly, for a vector $\mathsf{b} \in \mathbb{R}^s$, we define $\mathsf{b}(\mathbf{i}) \in \mathbb{R}^s$ as the vector whose lth entry is the i_lth entry of b.

2 METHOD

Conceptually, the proposed bootstrap method for estimating $\varepsilon_{1-\alpha}$ is based on generating a collection of "pseudo error variables" $\varepsilon_1^\star, \dots, \varepsilon_N^\star$ that behave approximately like i.i.d. samples of the unknown error variable $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\diamond$. Once the pseudo error variables have been generated, their empirical $(1-\alpha)$-quantile can then be used to define the estimate $\tilde{\varepsilon}_{1-\alpha}$. For example, if the user chooses $\alpha = 0.1$ and $N = 100$, then the estimate $\tilde{\varepsilon}_{1-\alpha}$ is defined as the 90th percentile among $\varepsilon_1^\star, \dots, \varepsilon_{100}^\star$.

The subtlety of this approach consists in finding an effective way to generate $\varepsilon_1^\star, \dots, \varepsilon_N^\star$. As a heuristic, we can imagine generating a random matrix $\widetilde{\mathsf{K}}^\star$ such that the difference $(\widetilde{\mathsf{K}}^\star - \widetilde{\mathsf{K}})$ is "statistically similar" to the difference $(\widetilde{\mathsf{K}} - \mathsf{K})$, and then defining each $\varepsilon_j^\star$ to be of the form $\|\widetilde{\mathsf{K}}^\star - \widetilde{\mathsf{K}}\|_\diamond$.

To explain this in more detail, it is important to notice that the matrix Z used to define $\widetilde{\mathsf{K}}$ has two special properties: (1) The columns of Z are i.i.d. (2) The columns of Z are generated so that $\mathbf{E}[\mathsf{Z}\mathsf{Z}^\top] = \mathsf{K}$. Accordingly, we can try to generate an analogous matrix $\mathsf{Z}^\star$ having columns that are conditionally i.i.d. given Z, and satisfying the conditional expectation relation $\mathbf{E}[\mathsf{Z}^\star(\mathsf{Z}^\star)^\top | \mathsf{Z}] = \widetilde{\mathsf{K}}$. Then, we can let $(\mathsf{Z}^\star(\mathsf{Z}^\star)^\top - \mathsf{Z}\mathsf{Z}^\top)$ play the role of the matrix $(\widetilde{\mathsf{K}}^\star - \widetilde{\mathsf{K}})$ mentioned earlier, and define pseudo error variables $\varepsilon_j^\star$ having the form $\|\mathsf{Z}^\star(\mathsf{Z}^\star)^\top - \mathsf{Z}\mathsf{Z}^\top\|_\diamond$. Furthermore, it turns out that these desired characteristics of $\mathsf{Z}^\star$ can be achieved by sampling its columns with replacement from the columns of Z, which leads to the formulation of Algorithm 1 below.

In settings where RFF approximation error is measured in terms of $|\psi(\tilde{k}) - \psi(k)|$, the principles just discussed carry over analogously, and Algorithm 1 provides corresponding pseudo error variables $\delta_1^\star, \dots, \delta_N^\star$.

Algorithm 1. (Error estimation for RFF)

Input: A positive integer N, a number $\alpha \in (0,1)$, the matrix of random features $\mathsf{Z} \in \mathbb{R}^{n \times s}$, and the random functions $Z_1(\cdot), \dots, Z_s(\cdot)$.

For: $j = 1, \dots, N$ **do in parallel**

- Draw a random vector $\mathbf{i} = (i_1, \dots, i_s)$ by sampling s numbers with replacement from $\{1, \dots, s\}$.
- Define the $n \times s$ matrix $\mathsf{Z}^\star = \mathsf{Z}(:, \mathbf{i})$.
- Define the function
$$\tilde{k}^\star(\cdot, \cdot') = \frac{1}{s}\Big(Z_{i_1}(\cdot)Z_{i_1}(\cdot') + \cdots + Z_{i_s}(\cdot)Z_{i_s}(\cdot')\Big).$$
- Compute the pseudo error variables
$$\varepsilon_j^\star := \|\mathsf{Z}^\star(\mathsf{Z}^\star)^\top - \mathsf{Z}\mathsf{Z}^\top\|_\diamond \quad \text{and} \quad \delta_j^\star := |\psi(\tilde{k}^\star) - \psi(\tilde{k})|.$$

Return: The estimates $\tilde{\varepsilon}_{1-\alpha} :=$ quantile$(\varepsilon_1^\star, \dots, \varepsilon_N^\star; 1-\alpha)$ and $\tilde{\delta}_{1-\alpha} :=$ quantile$(\delta_1^\star \dots, \delta_N^\star; 1-\alpha)$.

Remarks. There are a few basic aspects of Algorithm 1 that are helpful to note for practical purposes. First, it is not always necessary to explicitly form the matrix $\mathsf{Z}^\star(\mathsf{Z}^\star)^\top - \mathsf{Z}\mathsf{Z}^\top$, and this will be explained in greater detail in Section 3. Second, the matrix $\mathsf{Z}^\star$ and function $\tilde{k}^\star$ can be overwritten after each iteration, which is why they are not marked with a subscript j. Third, the number of bootstrap iterations N generally does not need to be very large, and our experiments in Section 5 illustrate that $N \sim 50$ is often sufficient for a variety of tasks.

3 COMPUTATIONAL EFFICIENCY

This section highlights the computational merits of Algorithm 1, and describes techniques for accelerating both error estimation and RFF. Since most of the ideas apply equally well to estimating both types of error, $\varepsilon_{1-\alpha}$ and $\delta_{1-\alpha}$, we mainly address the former.

3.1 Selecting the number of features by extrapolation

In the literature on bootstrap methods, a classical approach to speeding up computations is through the use of extrapolation techniques (Bickel and Yahav, 1988). In our current setting, this approach can be adapted as a two-step process: In the first step, we estimate the error of a "preliminary" RFF approximation that is computed from a small number of random features, say s_0. In the second step, we use this error estimate to predict how much $\varepsilon_{1-\alpha}$ will decrease with a larger number of features, say $s_1 \gg s_0$. More concretely, if we make the dependence of $\varepsilon_{1-\alpha}$ and $\tilde{\varepsilon}_{1-\alpha}$ on a generic value of s explicit by writing $\varepsilon_{1-\alpha}(s)$ and $\tilde{\varepsilon}_{1-\alpha}(s)$, then we seek to estimate $\varepsilon_{1-\alpha}(s_1)$ by extrapolating from $\tilde{\varepsilon}_{1-\alpha}(s_0)$.

There are two key benefits of this type of extrapolation. First, it can substantially speed up the error estimation process, because extrapolation only relies on $\tilde{\varepsilon}_{1-\alpha}(s_0)$, which is computed by running Algorithm 1 on a small instance of Z with size $n \times s_0$. (If extrapolation is not used, then computing $\tilde{\varepsilon}_{1-\alpha}(s_1)$ requires a much larger instance of Z with size $n \times s_1$.) Second, extrapolation enhances RFF by enabling the user to choose a value of s_1 that is "just large enough" so that $\varepsilon_{1-\alpha}(s_1)$ nearly matches a preferred error tolerance. In other words, this avoids the wasted computation that occurs when a user selects a highly excessive number of features due to uncertainty about accuracy.

From an algorithmic standpoint, an extrapolation rule can be developed as follows. Since it is possible to write $(\widetilde{\mathsf{K}} - \mathsf{K})$ as a sample average of s independent and zero-mean random matrices, the central limit theorem suggests heuristically that $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\diamond$ should decrease stochastically like $1/\sqrt{s}$ as a function of s. This also suggests that $\varepsilon_{1-\alpha}(s_1)$ should be smaller than $\varepsilon_{1-\alpha}(s_0)$ by a factor of $\sqrt{s_0/s_1}$, and so we define the extrapolated estimate of $\varepsilon_{1-\alpha}(s_1)$ as

$$\tilde{\varepsilon}^{\,\mathrm{EXT}}_{1-\alpha}(s_1) \;=\; \sqrt{\tfrac{s_0}{s_1}}\,\tilde{\varepsilon}_{1-\alpha}(s_0). \tag{3.1}$$

Hence, if a user wants to select s_1 so that $\varepsilon_{1-\alpha}(s_1) = \varepsilon_{\mathrm{tol}}$ for some tolerance $\varepsilon_{\mathrm{tol}}$, then s_1 can be chosen by setting $\tilde{\varepsilon}^{\,\mathrm{EXT}}_{1-\alpha}(s_1) = \varepsilon_{\mathrm{tol}}$, which leads to the choice $s_1 = s_0(\tilde{\varepsilon}_{1-\alpha}(s_0)/\varepsilon_{\mathrm{tol}})^2$. In Section 5, our experiments illustrate the effectiveness of this rule when s_1 is larger than s_0 by *two orders of magnitude*, demonstrating that extrapolation can yield major computational savings.

3.2 Low communication and parallel processing

In modern computing environments, communication costs are often of greater concern than processing costs (Martinsson and Tropp, 2020, §16.2). For this reason, it is important to emphasize that when $\varepsilon_{1-\alpha}$ is being estimated, Algorithm 1 *does not require any access* to the $n \times n$ matrices K or $\widetilde{\mathsf{K}}$, but only to the much smaller matrix Z. In fact, when extrapolation is used, Algorithm 1 only needs access to a "preliminary" instance of Z with s_0 columns, rather than a "full" instance of Z with $s_1 \gg s_0$ columns that will be used for a high-quality RFF approximation.

Another valuable feature of Algorithm 1 is its "embarrassingly parallel" structure. This means that the N iterations of the for-loop can be trivially distributed across a collection of, say m, processors. Furthermore, our experiments will demonstrate that $N \sim 50$ is sufficient in many situations, and so if the user has access to just one or two dozen processors, it is often realistic to treat the number of bootstrap iterations per processor N/m as being $\mathcal{O}(1)$.

3.3 Computational cost in illustrative cases

In this subsection, we quantify the computational cost of Algorithm 1 in some specific cases, with the benefits of extrapolation and parallel processing taken into account. The overall point of these examples is to show that the added cost of error estimation is manageable in comparison to the typical cost of RFF itself. As a benchmark for comparisons, it is worth noting that common learning tasks performed with RFF, such as kernel PCA and kernel ridge regression, have costs that are $\mathcal{O}(s_1^2 n)$ (Lopez-Paz et al., 2014; Avron et al., 2017). (Here and below, we continue to use s_0 and s_1 respectively to denote number of features used for preliminary and high-quality RFF approximations.)

Kernel matrix approximation. First, we consider the cost of computing $\tilde{\varepsilon}^{\,\mathrm{EXT}}_{1-\alpha}$ when error is measured through the operator norm $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_{\mathrm{op}}$. Importantly, the matrix $\mathsf{Z}^\star(\mathsf{Z}^\star)^\top - \mathsf{Z}\mathsf{Z}^\top$ in Algorithm 1 does not need to be explicitly formed when computing each $\varepsilon_j^\star$. The reason is that the norm $\|\mathsf{Z}^\star(\mathsf{Z}^\star)^\top - \mathsf{Z}\mathsf{Z}^\top\|_{\mathrm{op}}$ can be computed with variants of the power method, whose iterations are based on matrix-vector products $\mathsf{Z}^\star[(\mathsf{Z}^\star)^\top v] - \mathsf{Z}[\mathsf{Z}^\top v]$ with $v \in \mathbb{R}^n$ (Golub and Van Loan, 2013). Also, as a basic guideline, the number of power iterations may be taken as $\mathcal{O}(\log(n))$ (Martinsson and Tropp, 2020, §6.2.3). In this case, each iteration of Algorithm 1 incurs a cost of $\mathcal{O}(s_0 n \log(n))$. Hence, if the iterations are computed in parallel, and the number of iterations per processor is $N/m = \mathcal{O}(1)$ (as described above), then the overall runtime is $\mathcal{O}(s_0 n \log(n))$. Altogether, this compares well with the benchmark cost of $\mathcal{O}(s_1^2 n)$ when $s_1 \gg s_0$.

Alternatively, there is a second way to compute the norm $\|\mathsf{Z}^\star(\mathsf{Z}^\star)^\top - \mathsf{Z}\mathsf{Z}^\top\|_{\mathrm{op}}$ with lower communication costs. This approach originates from ideas in Epperly and Tropp (2022) and is based on computing a QR factorization $\mathsf{Z} = \mathsf{Q}\mathsf{R}$, where $\mathsf{Q} \in \mathbb{R}^{n \times s_0}$ has orthonormal columns, and $\mathsf{R} \in \mathbb{R}^{s_0 \times s_0}$ is upper-triangular. By noting that the relation $\mathsf{Z}(:, \mathbf{i}) = \mathsf{Q}(\mathsf{R}(:, \mathbf{i}))$ holds for every index vector $\mathbf{i}$ appearing in Algorithm 1, it follows from the unitary invariance of the

operator norm that

$$\|(\mathsf{Z}^\star)(\mathsf{Z}^\star)^\top - \mathsf{Z}\mathsf{Z}^\top\|_{\text{op}} = \|\mathsf{R}(:,\mathbf{i})\mathsf{R}(:,\mathbf{i})^\top) - \mathsf{R}\mathsf{R}^\top\|_{\text{op}}. \quad (3.2)$$

So, if the bootstrap iterations are distributed across many processors, then the identity (3.2) shows that it is only necessary to communicate copies of the $s_0 \times s_0$ matrix R to the processors, rather than copies of the $n \times s_0$ matrix Z. Also, the cost of computing the right side of (3.2) at each iteration is only $\mathcal{O}(s_0^2 \log(s_0))$. However, these gains are offset by a one-time cost of $\mathcal{O}(s_0^2 n)$ that must be paid to extract R. In the case when $N/m = \mathcal{O}(1)$, this approach leads to an overall runtime of $\mathcal{O}(s_0^2 n)$. Although this nominally exceeds the $\mathcal{O}(s_0 n \log(n))$ runtime of the previous approach when $\log(n) = \mathcal{O}(s_0)$, the reduced communication of this approach might still lead to better performance in practice. Also, this approach is favorable when parallel processing is limited, because its cost per iteration is lower.

Kernel ridge regression. As our second illustration of computational cost, we consider the use of Algorithm 1 in estimating the extra mean-squared test error that arises from RFF in kernel ridge regression.

However, before diving into the details of cost, we first review the basic elements of kernel ridge regression and its associated RFF approximation. For a kernel k, let f_k denote a kernel ridge regression function trained on n data points in $\mathbb{R}^d$. This means that if the training points are denoted as $(x_1, y_1), \ldots, (x_n, y_n) \in \mathbb{R}^d \times \mathbb{R}$ with $\mathsf{y} = (y_1, \ldots, y_n)$, then

$$f_k(\cdot) = \sum_{i=1}^{n} \beta_i k(x_i, \cdot), \quad (3.3)$$

where the vector $\beta \in \mathbb{R}^n$ solves $(\mathsf{K} + \lambda \mathsf{I}_n)\beta = \mathsf{y}$, and $\lambda > 0$ is a tuning parameter. For the RFF approximation $\tilde{k}$, the associated regression function is defined as

$$f_{\tilde{k}}(\cdot) = \sum_{i=1}^{s} \tilde{\beta}_i Z_i(\cdot), \quad (3.4)$$

where the vector $\tilde{\beta} \in \mathbb{R}^s$ solves $(\mathsf{Z}^\top \mathsf{Z} + \lambda \mathsf{I}_s)\tilde{\beta} = \mathsf{Z}^\top \mathsf{y}$ and the functions $Z_1(\cdot), \ldots, Z_s(\cdot)$ are as defined in Section 1.1. Next, let $\psi(k)$ denote the mean-squared test error of f_k. More specifically, if there are t test points denoted as $(x_1', y_1'), \ldots, (x_t', y_t') \in \mathbb{R}^d \times \mathbb{R}$, then we have

$$\psi(k) = \frac{1}{t} \sum_{i=1}^{t} (y_i' - f_k(x_i'))^2. \quad (3.5)$$

Likewise, let $\psi(\tilde{k})$ denote the corresponding quantity involving $f_{\tilde{k}}$.

Returning our attention to error estimation, let $\delta_{1-\alpha}$ denote the $(1-\alpha)$-quantile of $\psi(\tilde{k}) - \psi(k)$. Our goal here is to quantify the cost of computing an extrapolated estimate $\tilde{\delta}_{1-\alpha}^{\text{EXT}}$ for $\delta_{1-\alpha}$. In this particular setting, there are a few ways to reduce the cost of Algorithm 1 by doing some one-time computations before starting the for-loop. Namely, it is helpful to compute the scalar value $\psi(\tilde{k})$, as well as the vector $\mathsf{b} = \mathsf{Z}^\top \mathsf{y}$, and the QR factorization $\mathsf{Z} = \mathsf{Q}\mathsf{R}$. (The motivation for the QR factorization is similar to that discussed earlier in connection with the work of Epperly and Tropp (2022).)

Inside the for-loop, each iteration computes a separate instance of the pseudo error variable $\psi(\tilde{k}^\star) - \psi(\tilde{k})$, with $\tilde{k}^\star$ being as defined in Algorithm 1. Since $\psi(\tilde{k})$ has been pre-computed, it is only necessary to compute $\psi(\tilde{k}^\star)$. This requires computing the solution $\tilde{\beta}^\star \in \mathbb{R}^{s_0}$ of the equation $((\mathsf{Z}^\star)^\top(\mathsf{Z}^\star) + \lambda \mathsf{I}_{s_0})\tilde{\beta}^\star = \mathsf{b}^\star$, where $\mathsf{Z}^\star = \mathsf{Z}(:,\mathbf{i})$ and $\mathsf{b}^\star = \mathsf{b}(\mathbf{i})$. But instead of solving this equation directly, the initial QR factorization allows it to be solved more efficiently as $\big(\mathsf{R}(:,\mathbf{i})^\top \mathsf{R}(:,\mathbf{i}) + \lambda \mathsf{I}_{s_0}\big)\tilde{\beta}^\star = \mathsf{b}^\star$. Once the solution $\tilde{\beta}^\star$ is in hand, the scalar $\psi(\tilde{k}^\star)$ can be computed similarly to (3.5), by replacing f_k with $f_{\tilde{k}^\star}(\cdot) = \tilde{\beta}_1^\star Z_{i_1}(\cdot) + \cdots + \tilde{\beta}_s^\star Z_{i_{s_0}}(\cdot)$, where it should be noted that the subscripts $i_1, \ldots, i_{s_0}$ are the entries of $\mathbf{i}$.

To arrive at a simple overall runtime for computing $\tilde{\delta}_{1-\alpha}^{\text{EXT}}$, suppose the for-loop is distributed so that the number of iterations per processor satisfies $N/m = \mathcal{O}(1)$. In addition, suppose that the number of test points satisfies $t = \mathcal{O}(n)$, and the data dimension satisfies $d = \mathcal{O}(s_0)$. Under these assumptions, the overall runtime to compute $\tilde{\delta}_{1-\alpha}^{\text{EXT}}$, is $\mathcal{O}(s_0^2 n)$, which is quite manageable in comparison to the $\mathcal{O}(s_1^2 n)$ cost of kernel ridge regression using RFF.

4 THEORY

Here, we analyze the performance of Algorithm 1 when the RFF kernel approximation error is measured in a uniform entrywise sense, which is common in the literature (e.g. Rahimi and Recht, 2007; Sutherland and Schneider, 2015; Liu et al., 2021). In particular, we use the norm $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\infty = \max_{1 \le j,j' \le n} |\widetilde{\mathsf{K}}_{jj'} - \mathsf{K}_{jj'}|$. Our main theoretical result shows that in the limit of large problem sizes ($n \to \infty$), the estimate $\tilde{\varepsilon}_{1-\alpha}$ constructed in Algorithm 1 matches the performance of the ideal value $\varepsilon_{1-\alpha}$ with respect to coverage probability.

Assumptions. We consider a sequence of kernel approximation problems indexed by $n = 1, 2, \ldots$, where the dimension $d = d_n$ of the point set $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ is allowed to vary in an unrestricted manner as $n \to \infty$. In addition, the kernel function $k = k_n$ may vary as $n \to \infty$, provided that it is of the type described in Section 1.1. That is, the kernel function k is assumed to be shift-invariant and continuous with $k(0,0) = 1$ for every n.

With regard to RFF and error estimation, the number of random features $s = s_n$ and bootstrap iterations $N = N_n$

in Algorithm 1 are both allowed to vary as $n \to \infty$, subject to two basic conditions: $N \to \infty$ and $\frac{\log(n)^5}{s} \to 0$.

Theorem 1. *Suppose that the aforementioned assumptions hold. Also let $\tilde{\varepsilon}_{1-\alpha}$ be computed with Algorithm 1 from an input matrix $\mathsf{Z} \in \mathbb{R}^{n \times s}$ that is generated as described in Section 1.1. Then, for any fixed $\alpha \in (0,1)$, as $n \to \infty$,*

$$\mathbf{P}\Big(\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\infty \le \tilde{\varepsilon}_{1-\alpha} \Big) \to 1 - \alpha. \tag{4.1}$$

Remarks. Theorem 1 has been presented in an asymptotic form for the sake of simplicity. An explicit rate of convergence can be found in the proof in Appendix A, which shows that the probability in (4.1) differs from $(1-\alpha)$ by a quantity that is at most $\mathcal{O}\big((\log(N)/N)^{1/2} + (\log(sn)^5/s)^{1/4}\big)$. To interpret some other aspects of the result, it should be emphasized that the assumptions are mild, insofar far as the point set $\{x_1, \dots, x_n\} \subset \mathbb{R}^d$ is unrestricted with respect to its geometric structure and dimension d. Also, there are no extra assumptions on the kernel function beyond those that are ordinarily used in the study of RFF. Furthermore, the conditions on N and s are mild, since they allow both N and s to grow very slowly compared to n. On the other hand, to note a limitation of Theorem 1, it only deals with the typical version of RFF where the columns of the random matrix Z are independent (as in Section 1.1), and it does not cover some particular versions of RFF in which these columns may not be independent (Le et al., 2013; Choromanski and Sindhwani, 2016). However, even in the typical setting, our proof utilizes cutting-edge results on the central limit theorem in high dimensions (Chernozhuokov et al., 2022), and the challenge extending such results to account for dependence is at the frontier of research in high-dimensional probability.

5 EXPERIMENTS

We demonstrate the empirical performance of our error estimates in three settings: kernel matrix approximation (Section 5.1), kernel ridge regression (Section 5.2), and kernel-based hypothesis testing (Appendix B). There are two main takeaways: First, the extrapolated estimates $\tilde{\varepsilon}^{\text{EXT}}_{1-\alpha}$ and $\tilde{\delta}^{\text{EXT}}_{1-\alpha}$ closely track their targets $\varepsilon_{1-\alpha}$ and $\delta_{1-\alpha}$ across different settings. Second, these estimates can be quickly computed with modest values of s_0 and N. A Python implementation of the experiments is available at the GitHub repository Yao et al. (2023).

5.1 Error estimation for RFF in kernel matrix approximation

Here, we examine how accurate $\tilde{\varepsilon}_{1-\alpha}$ and $\tilde{\varepsilon}^{\text{EXT}}_{1-\alpha}$ are as estimates of $\varepsilon_{1-\alpha}$. This is done when matrix approximation error is measured through the ℓ_∞-norm $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\infty$ (Figure 2), as well as the operator norm $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_{\text{op}}$ (Figure 3).

Data examples. The results are based on two datasets derived from: (1) the *Lorenz system* (Lorenz, 1963) and (2) the training set portion of MNIST (LeCun et al., 1998). The Lorenz system is a well-known chaotic dynamical system, and we followed (Erichson et al., 2018) by generating $n = 25000$ points that reside on a trajectory in $\mathbb{R}^3$. The training set portion of MNIST consists of $n = 50000$ points that represent 784-pixel images.

Design of experiments. The following procedures were used for both datasets, with the kernel matrix $\mathsf{K} \in \mathbb{R}^{n \times n}$ being computed directly from the data. For each value of s in a grid ranging from 50 to 6000, we generated 300 realizations of the random matrix $\mathsf{Z} \in \mathbb{R}^{n \times s}$ as described in Section 1.1, using the probability distribution ρ corresponding to the Gaussian kernel $\exp(-\|x - x'\|_2^2/(2\sigma^2))$ with $\sigma \in \{0.5, 1, 4\}$. In addition, for each realization of Z, we computed the associated error variables $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\infty$ and $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_{\text{op}}$, where $\widetilde{\mathsf{K}} = \mathsf{Z}\mathsf{Z}^\top$. This provided us with a set of 300 realizations of each type of error variable, and we computed the 90th percentile of each set, treating it as ground truth for $\varepsilon_{0.9}$ at each s. In Figures 2 and 3, the value of $\varepsilon_{0.9}$ at each s is plotted with a black curve. To ease comparisons, the black curve was rescaled so that its initial value is 1 in each plot, and the associated blue and red curves (described below) were rescaled by the same factor.

Next, we applied Algorithm 1 with $N = 30$ iterations to each realization of Z, yielding 300 corresponding estimates $\tilde{\varepsilon}_{0.9}$ at each s, and we plotted the average of these estimates with a blue curve. Also, from each of the 300 realizations of $\tilde{\varepsilon}_{0.9}$ computed at $s_0 = 50$, we obtained the extrapolated estimates $\tilde{\varepsilon}^{\text{EXT}}_{0.9}(s)$ using formula (3.1) for all $50 \le s \le 6000$. The average of the extrapolated estimates is plotted with a red curve, and a pink envelope signifies ± 1 standard deviation. (Note that in some plots within Figure 2, the pink envelope is almost entirely covered by the red curve.)

Discussion of results. It is clear that both the blue and red curves for $\tilde{\varepsilon}_{0.9}$ and $\tilde{\varepsilon}^{\text{EXT}}_{0.9}$ closely track the black curve representing ground truth. Beyond this main point, the red curve deserves special attention—because it is based on extrapolation from only $s_0 = 50$ features. So, if the user constructs a "preliminary" kernel approximation with 50 features, they can use Algorithm 1 to "look ahead" and accurately predict how error will decrease for larger choices of s, e.g. up to $s = 6000$. Computationally, this means Algorithm 1 can be run with a matrix Z that is $n \times 50$, rather than $n \times 6000$ for a non-extrapolated estimate, i.e. *two orders of magnitude reduction*. Another important point is that the number of bootstrap iterations $N = 30$ is so small that, with a dozen processors, only a few iterations are needed per processor. Lastly, the two figures show that

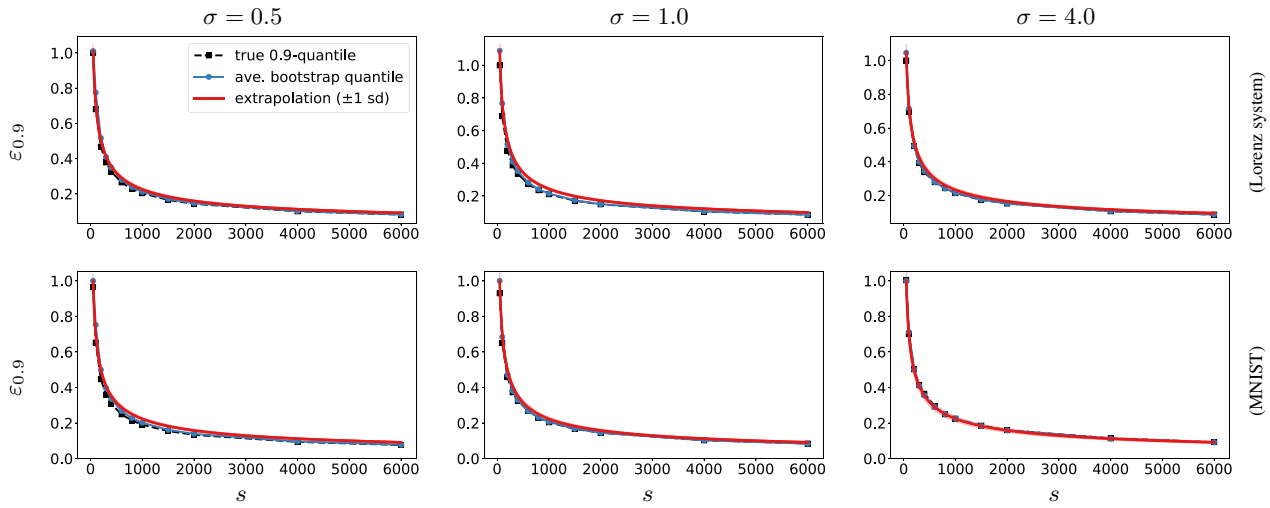

Figure 2: (Estimation of $\varepsilon_{0.9}$ for $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_{\infty}$.) The top and bottom rows correspond respectively to the Lorenz system and MNIST datasets. The columns correspond to choices of the kernel bandwidth σ.

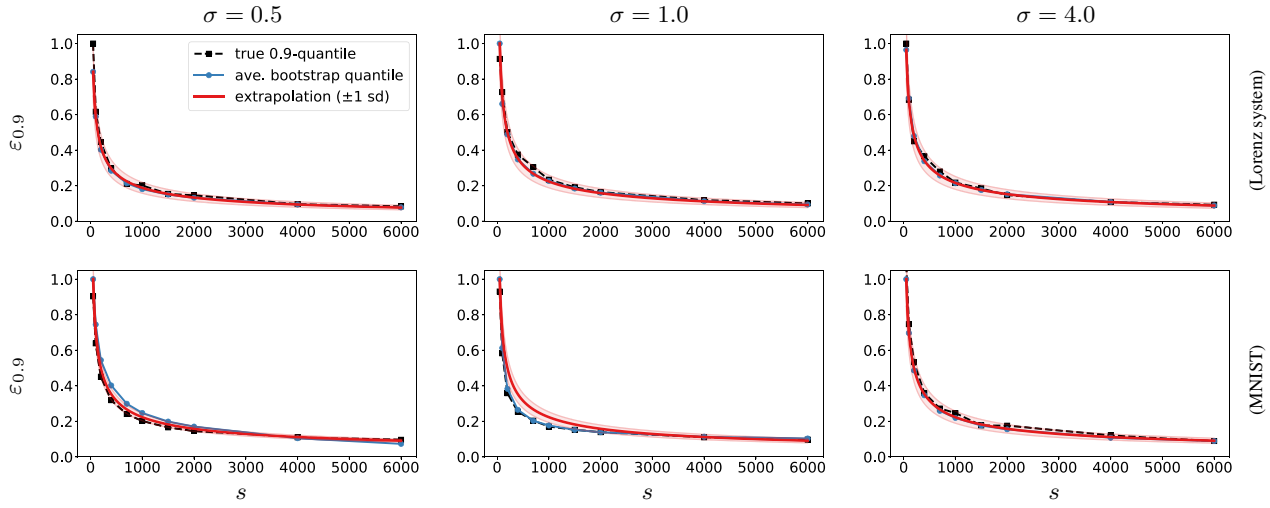

Figure 3: (Estimation of $\varepsilon_{0.9}$ for $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_{\mathrm{op}}$.) The plots are organized analagously to Figure 2.

the error estimates behave reliably across different norms, datasets, and bandwidths.

5.2 Error estimation for RFF in kernel ridge regression

Now we turn our attention to estimating how much error is created by RFF in kernel ridge regression.

Data examples. We used two regression datasets, each consisting of (x, y) pairs in $\mathbb{R}^d \times \mathbb{R}$ with $d = 50$. Each dataset $\mathcal{D}$ was partitioned as $\mathcal{D} = \mathcal{D}_{\text{train}} \cup \mathcal{D}_{\text{test}}$, with $|\mathcal{D}_{\text{test}}| = 3000$ and $n = |\mathcal{D}_{\text{train}}| = 27000$. To obtain two different versions of $\mathcal{D}$ with these specifications, we uniformly subsampled 30000 rows and 50 columns from the datasets *YearPredictionMSD* and *Buzz in social media* in the repository (Dua and Graff, 2017). For both versions of $\mathcal{D}$, we applied the standard normalization function 'MinMaxScaler' from scikit-learn to all the x vectors, and in the case of YearPredictionMSD we took the square-root of the y values due to their wide range.

Design of experiments. For a kernel k, let $\psi(k)$ denote the mean-squared test error of the associated ridge regression function, as defined in (3.5). Also, let $\delta_{0.9}$ denote the 90th percentile of the random variable $\psi(\tilde{k}) - \psi(k)$, which measures the extra prediction error due to RFF. The experiments here were organized analogously to those in Section 5.1, with $(\delta_{0.9}, \tilde{\delta}_{0.9}, \tilde{\delta}^{\text{EXT}}_{0.9})$ playing the roles of $(\varepsilon_{0.9}, \tilde{\varepsilon}_{0.9}, \tilde{\varepsilon}^{\text{EXT}}_{0.9})$. Hence, the colored curves and the envelope can be interpreted in the same way. Also, as before, we generated 300 realizations of Z and used $N = 30$ at each value of s. There are only a few notable details that are specific to the current setting. First, we computed $\tilde{\delta}^{\text{EXT}}_{0.9}$ by extrapolating from the initial value $s_0 = 200$, and we always fixed the regression

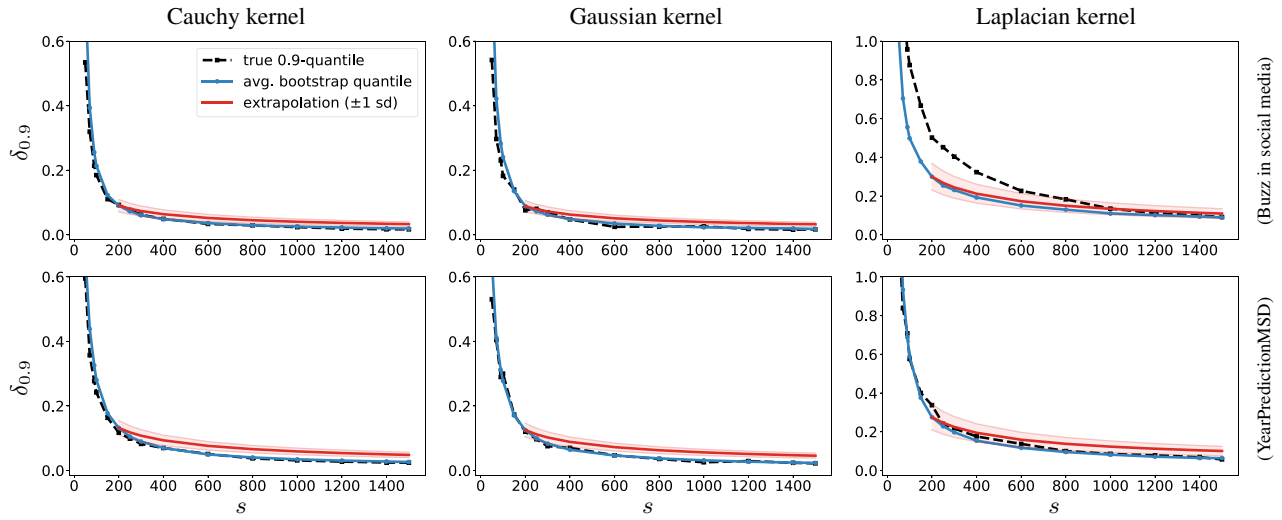

Figure 4: (Estimation of $\delta_{0.9}$ for $\psi(\tilde{k}) - \psi(k)$.) The top and bottom rows correspond respectively to the two regression datasets. The columns correspond to the three different kernels.

tuning parameter at $\lambda = 1$. Second, all the curves were multiplied by the number $1/\psi(k)$ so that they can be more naturally viewed on a scale relative to the mean-squared test error of f_k. Third, we performed the experiments using three different kernels: the Gaussian kernel $\exp(-\|x - x'\|_2^2/10)$, the Laplacian kernel $\exp(-\|x - x'\|_1/10)$, and the Cauchy kernel $\prod_{j=1}^{d} 1/(1 + \Delta_j^2/10)$ where $\Delta = x - x'$.

Discussion of results. Figure 4 shows that in kernel ridge regression, the error estimates $\tilde{\delta}_{0.9}$ and $\tilde{\delta}_{0.9}^{\text{EXT}}$ perform well, and with qualitatively similar characteristics to the error estimates in Section 5.1. However, this setting is more challenging, since a larger value of $s_0 = 200$ is needed, and since $\tilde{\delta}_{0.9}^{\text{EXT}}$ shows a slight upward bias for large s. Nevertheless, an upward bias may be preferred as being safer than a downward bias in the context of error estimation. In addition, Figure 4 shows that the error estimates largely maintain their accuracy across different choices of kernels.

6 CONCLUSION

Despite the broad impact that RFF has had in scaling up kernel methods, a longstanding difficulty for users is that they do not know the actual errors of RFF approximations. This paper offers the first systematic approach to numerically estimate these errors. Our approach also overcomes practical limitations of analytical worst-case error bounds, because the error estimates are tailored to the user's specific inputs, and are very flexible with respect to different problem settings and error metrics. Computationally, our approach leverages both parallelism and extrapolation so that the additional step of error estimation is affordable in relation to RFF itself. Also, our approach can enhance the efficiency of RFF by guiding the user to select s in a data-adaptive way. From the standpoint of theory, we have provided a guarantee in the context of kernel matrix approximation, showing that our error estimates perform properly under mild assumptions. Furthermore, we have demonstrated empirically that our error estimates are quite accurate in a variety of tasks.

Looking ahead to future work, it is important to recognize that there are many variants and uses of RFF that go beyond the setup considered here. For example, our approach might be adapted to settings involving rotation-invariant kernels (Lyu, 2017; Choromanski et al., 2017), low-precision and quantized kernel estimators (Zhang et al., 2019; Li and Li, 2021), or random features that are not independent (Le et al., 2013; Choromanski and Sindhwani, 2016).

Acknowledgements

MEL gratefully acknowledges partial support from NSF grant DMS-1915786. NBE would like to acknowledge partial support from the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Scientific Discovery through Advanced Computing (SciDAC) program, under Contract Number DE-AC02-05CH11231, and the National Energy Research Scientific Computing Center (NERSC), operated under Contract No. DE-AC02-05CH11231 at Lawrence Berkeley National Laboratory. The authors thank all the reviewers for their helpful and constructive feedback.

References

Ahfock, D. C., Astle, W. J., and Richardson, S. (2021). Statistical properties of sketching algorithms. *Biometrika*, 108(2):283–297.

Ainsworth, M. and Oden, J. T. (2011). *A Posteriori Error*

Estimation in Finite Element Analysis, volume 37. John Wiley & Sons.

Avron, H., Kapralov, M., Musco, C., Musco, C., Velingker, A., and Zandieh, A. (2017). Random Fourier features for kernel ridge regression: Approximation bounds and statistical guarantees. In *International Conference on Machine Learning*.

Babuška, I. and Rheinboldt, W. C. (1978). Error estimates for adaptive finite element computations. *SIAM Journal on Numerical Analysis*, 15(4):736–754.

Bank, R. E. and Weiser, A. (1985). Some a posteriori error estimators for elliptic partial differential equations. *Mathematics of Computation*, 44(170):283–301.

Bickel, P. J. and Yahav, J. A. (1988). Richardson extrapolation and the bootstrap. *Journal of the American Statistical Association*, 83(402):387–393.

Chernozhuokov, V., Chetverikov, D., Kato, K., and Koike, Y. (2022). Improved central limit theorem and bootstrap approximations in high dimensions. *The Annals of Statistics*, 50(5):2562 – 2586.

Choromanski, K. and Sindhwani, V. (2016). Recycling randomness with structure for sublinear time kernel expansions. In *International Conference on Machine Learning*.

Choromanski, K. M., Rowland, M., and Weller, A. (2017). The unreasonable effectiveness of structured random orthogonal embeddings. In *Advances in Neural Information Processing Systems*.

Dai, B., Xie, B., He, N., Liang, Y., Raj, A., Balcan, M.-F. F., and Song, L. (2014). Scalable kernel methods via doubly stochastic gradients. In *Advances in Neural Information Processing Systems*.

Davison, A. C. and Hinkley, D. V. (1997). *Bootstrap Methods and Their Application*. Cambridge.

Dua, D. and Graff, C. (2017). UCI machine learning repository.

Epperly, E. N. and Tropp, J. A. (2022). Jackknife variability estimation for randomized matrix computations. *arXiv:2207.06342*.

Erichson, N. B., Mathelin, L., Brunton, S. L., and Kutz, J. N. (2018). Diffusion maps meet Nyström. *arXiv:1802.08762*.

Giannakis, D., Henriksen, A., Tropp, J. A., and Ward, R. (2022). Learning to forecast dynamical systems from streaming data. *SIAM Journal on Applied Dynamical Systems*.

Golub, G. H. and Van Loan, C. F. (2013). *Matrix Computations*. JHU Press.

Gretton, A., Borgwardt, K. M., Rasch, M. J., Schölkopf, B., and Smola, A. (2012). A kernel two-sample test. *Journal of Machine Learning Research*, 13(1):723–773.

Halko, N., Martinsson, P.-G., and Tropp, J. A. (2011). Finding structure with randomness: Probabilistic algorithms for constructing approximate matrix decompositions. *SIAM review*, 53(2):217–288.

Hall, P. (2013). *The Bootstrap and Edgeworth Expansion*. Springer.

Kiessling, J., Ström, E., and Tempone, R. (2021). Wind field reconstruction with adaptive random Fourier features. *Proceedings of the Royal Society A*, 477(2255):20210236.

Le, Q., Sarlós, T., and Smola, A. (2013). Fastfood - Approximating kernel expansions in loglinear time. In *International Conference on Machine Learning*.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Li, X. and Li, P. (2021). Quantization algorithms for random Fourier features. In *International Conference on Machine Learning*.

Li, Z., Ton, J.-F., Oglic, D., and Sejdinovic, D. (2019). Towards a unified analysis of random Fourier features. In *International Conference on Machine Learning*, pages 3905–3914. PMLR.

Liberty, E., Woolfe, F., Martinsson, P.-G., Rokhlin, V., and Tygert, M. (2007). Randomized algorithms for the low-rank approximation of matrices. *Proceedings of the National Academy of Sciences*, 104(51):20167–20172.

Liu, F., Huang, X., Chen, Y., and Suykens, J. A. (2021). Random features for kernel approximation: A survey on algorithms, theory, and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):7128–7148.

Lopes, M. E. (2022). Central limit theorem and bootstrap approximation in high dimensions: Near $1/\sqrt{n}$ rates via implicit smoothing. *The Annals of Statistics*, 50(5):2492–2513.

Lopes, M. E., Erichson, N. B., and Mahoney, M. W. (2020). Error estimation for sketched SVD via the bootstrap. In *International Conference on Machine Learning*.

Lopes, M. E., Erichson, N. B., and Mahoney, M. W. (2023). Bootstrapping the operator norm in high dimensions: Error estimation for covariance matrices and sketching. *Bernoulli*, 29(1):428–450.

Lopes, M. E., Wang, S., and Mahoney, M. W. (2018). Error estimation for randomized least-squares algorithms via the bootstrap. In *International Conference on Machine Learning*.

Lopes, M. E., Wang, S., and Mahoney, M. W. (2019). A bootstrap method for error estimation in randomized matrix multiplication. *The Journal of Machine Learning Research*, 20(1):1434–1473.

Lopez-Paz, D., Sra, S., Smola, A., Ghahramani, Z., and Schölkopf, B. (2014). Randomized nonlinear component analysis. In *International Conference on Machine Learning*.

Lorenz, E. N. (1963). Deterministic nonperiodic flow. *Journal of Atmospheric Sciences*, 20(2):130–141.

Lunde, R., Sarkar, P., and Ward, R. (2021). Bootstrapping the error of Oja's algorithm. In *Advances in Neural Information Processing Systems*.

Lyu, Y. (2017). Spherical structured feature maps for kernel approximation. In *International Conference on Machine Learning*, pages 2256–2264. PMLR.

Marsland, S. (2011). *Machine Learning: An Algorithmic Perspective*. Chapman and Hall/CRC.

Martinsson, P.-G. and Tropp, J. A. (2020). Randomized numerical linear algebra: Foundations and algorithms. *Acta Numerica*, 29:403–572.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems*.

Rudi, A. and Rosasco, L. (2017). Generalization properties of learning with random features. In *Advances in Neural Information Processing Systems*.

Rudin, W. (1990). *Fourier Analysis on Groups*. Wiley.

Schölkopf, B. and Smola, A. J. (2002). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT.

Shao, J. and Tu, D. (2012). *The Jackknife and Bootstrap*. Springer.

Shawe-Taylor, J. and Cristianini, N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge.

Sriperumbudur, B. and Szabó, Z. (2015). Optimal rates for random Fourier features. In *Advances in Neural Information Processing Systems*.

Sun, Y., Gilbert, A., and Tewari, A. (2018). But how does it work in theory? Linear SVM with random features. In *Advances in Neural Information Processing Systems*.

Sutherland, D. J. and Schneider, J. (2015). On the error of random Fourier features. In *Conference on Uncertainty in Artificial Intelligence*.

van der Vaart, A. W. (2000). *Asymptotic Statistics*. Cambridge.

Verfürth, R. (1994). A posteriori error estimation and adaptive mesh-refinement techniques. *Journal of Computational and Applied Mathematics*, 50(1-3):67–83.

Woolfe, F., Liberty, E., Rokhlin, V., and Tygert, M. (2008). A fast randomized algorithm for the approximation of matrices. *Applied and Computational Harmonic Analysis*, 25(3):335–366.

Yang, T., Li, Y.-F., Mahdavi, M., Jin, R., and Zhou, Z.-H. (2012). Nyström method vs random Fourier features: A theoretical and empirical comparison. In *Advances in Neural Information Processing Systems*.

Yao, J., Erichson, N. B., and Lopes, M. E. (2023). `https://github.com/jwyyy/bootstrappedRFF`.

Zhang, J., May, A., Dao, T., and Ré, C. (2019). Low-precision random Fourier features for memory-constrained kernel approximation. In *International Conference on Artificial Intelligence and Statistics*.

Zhao, J. and Meng, D. (2015). FastMMD: Ensemble of circular discrepancy for efficient two-sample test. *Neural Computation*, 27(6):1345–1372.

Error Estimation for Random Fourier Features
Supplementary Material

The supplementary material consists of three appendices. Appendix A contains the proof of Theorem 1 from the main text. Appendix B presents empirical results on estimating the error of RFF in the context of kernel-based hypothesis testing. Appendix C is a continuation of Section 5.1 from the main text, and presents empirical results for an additional dataset in the context of kernel matrix approximation.

A Proof of Theorem 1

We begin by defining several distributions functions that will be needed throughout the proof. For any $t \in \mathbb{R}$, define

$$F_s(t) \;=\; \mathbf{P}\Big(\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\infty \leq t\Big), \tag{A.1}$$

$$\tilde{F}_s(t) \;=\; \mathbf{P}\big(\varepsilon_1^\star \leq t \,\big|\, \mathsf{Z}\big), \tag{A.2}$$

$$\tilde{F}_{s,N}(t) \;=\; \frac{1}{N}\sum_{j=1}^{N} 1\{\varepsilon_j^\star \leq t\}, \tag{A.3}$$

where $\varepsilon_1^\star, \ldots, \varepsilon_N^\star$ are generated as in Algorithm 1, and $1\{\cdot\}$ is an indicator function. Note also that $\tilde{F}_s$ and $\tilde{F}_{s,N}$ are random functions.

Below, we develop two lemmas showing that these distribution functions are uniformly close with high probability. The uniform approximations are important, because they imply that the quantiles of $\tilde{F}_{s,N}$ and F_s behave similarly—which is exactly what is needed to prove Theorem 1, since the $(1-\alpha)$-quantiles of $\tilde{F}_{s,N}$ and F_s are respectively $\tilde{\varepsilon}_{1-\alpha}$ and $\varepsilon_{1-\alpha}$.

Lemma A.1. *Suppose the conditions of Theorem 1 hold. Then, there is an absolute constant $c > 0$ such that the event*

$$\sup_{t\in\mathbb{R}} \big|\tilde{F}_{s,N}(t) - \tilde{F}_s(t)\big| \;\leq\; \tfrac{\sqrt{\log(N)}}{\sqrt{N}} \tag{A.4}$$

holds with probability at least $1 - c/N$.

Proof. Conditioning on Z, we can view $\tilde{F}_{s,N}$ as the empirical distribution function associated with N i.i.d. samples drawn from $\tilde{F}_s$. Consequently, the Dvoretzky-Kiefer-Wolfowitz inequality (van der Vaart, 2000, p.268) gives the following bound for any real number r,

$$\mathbf{P}\Big(\sup_{t\in\mathbb{R}} \big|\tilde{F}_{s,N}(t) - \tilde{F}_s(t)\big| > \tfrac{r}{\sqrt{N}} \,\Big|\, \mathsf{Z}\Big) \;\leq\; 2e^{-2r^2}. \tag{A.5}$$

Hence, the statement of the lemma follows by taking an expectation over Z and using the choice $r = \sqrt{\log(N)}$. □

Lemma A.2. *Suppose the conditions of Theorem 1 hold. Then, there is an absolute constant $c > 0$ such that the event*

$$\sup_{t\in\mathbb{R}} \big|\tilde{F}_s(t) - F_s(t)\big| \;\leq\; \Big(\tfrac{c\log(2sn)^5}{s}\Big)^{1/4} \tag{A.6}$$

holds with probability at least $1 - c(1/s + \sqrt{\log(sn)^3/s})$.

Proof. For each $i = 1, \ldots, s$, define a random matrix $\mathsf{Y}(i) \in \mathbb{R}^{n \times n}$ whose (j, j') entry is

$$\mathsf{Y}_{jj'}(i) \;=\; Z_i(x_j)Z_i(x_{j'}) \;-\; \mathbf{E}[Z_i(x_j)Z_i(x_{j'})], \tag{A.7}$$

and let $\bar{\mathsf{Y}} = \frac{1}{s}\sum_{i=1}^{s} \mathsf{Y}(i)$. Since the expectation of $Z_i(x_j)Z_i(x_{j'})$ is equal to $k(x_j, x'_j)$, we have

$$\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\infty \;=\; \max_{1 \le j, j' \le n} \big|\bar{\mathsf{Y}}_{jj'}\big|. \tag{A.8}$$

Next, let $\mathsf{Y}^\star(1), \ldots, \mathsf{Y}^\star(s)$ be i.i.d. samples with replacement from $(\mathsf{Y}(1), \ldots, \mathsf{Y}(s))$. Based on the definition of the bootstrap sample $\varepsilon_1^\star$ in Algorithm 1, it is straightforward to check that it can be expressed as

$$\varepsilon_1^\star \;=\; \max_{1 \le j, j' \le n} \Big|\tfrac{1}{s}\textstyle\sum_{i=1}^{s} \mathsf{Y}^\star_{jj'}(i) - \bar{\mathsf{Y}}_{jj'}\Big|. \tag{A.9}$$

Likewise, the left side of (A.6) satisfies

$$\sup_{t \in \mathbb{R}} |\tilde{F}_s(t) - F_s(t)| = \sup_{t \in \mathbb{R}} \left| \mathbf{P}\Big(\max_{1 \le j, j' \le n} \big|\tfrac{1}{s}\textstyle\sum_{i=1}^{s} \mathsf{Y}^\star_{jj'}(i) - \bar{\mathsf{Y}}_{jj'}\big| \le t \,\Big|\, \mathsf{Z}\Big) - \mathbf{P}\Big(\max_{1 \le j, j' \le n} |\bar{\mathsf{Y}}_{jj'}| \le t \Big) \right|.$$

Due to this representation and the fact that the matrices $\mathsf{Y}(1), \ldots, \mathsf{Y}(s)$ are i.i.d., the statement (A.6) follows as a consequence of Lemma 4.5 in (Chernozhuokov et al., 2022), provided that we can verify three conditions: Specifically, it is enough to show that there exist absolute constants $c_1, c_2, C > 0$ such that the following bounds (i), (ii), and (iii) hold for all $j, j' \in \{1, \ldots, n\}$,

(i) $\quad \mathrm{var}(\mathsf{Y}_{jj'}(1)) \;\ge\; c_1,$

(ii) $\quad \mathbf{E}[\mathsf{Y}^4_{jj'}(1)] \;\le\; C^2 c_2,$

(iii) $\quad \mathbf{E}[\exp(|\mathsf{Y}_{jj'}(1)|/C)] \;\le\; 2.$

As a first step toward verifying these conditions, note that the bound $|Z_1(x_j)| \le \sqrt{2}$ holds almost surely for all $j \in \{1, \ldots, n\}$ by construction. This implies $|\mathsf{Y}_{jj'}(1)| \le 4$ holds almost surely for all j, j', and so the existence of the two absolute constants $c_2, C > 0$ satisfying (ii) and (iii) is clear.

The only remaining item to address is the lower bound in condition (i). For this purpose, we begin by noting that

$$\begin{aligned} \mathrm{var}(\mathsf{Y}_{jj'}(1)) \;&=\; \mathrm{var}(Z_1(x_j)Z_1(x_{j'})) \\ &=\; \mathbf{E}\Big[\big(Z_1(x_j)Z_1(x'_j)\big)^2\Big] \;-\; k(x_j, x_{j'})^2. \end{aligned} \tag{A.10}$$

To handle the second moment in the last line, observe that the sum-of-angles identity $\cos(a)\cos(b) = \frac{1}{2}\cos(a-b) + \frac{1}{2}\cos(a+b)$ yields

$$\begin{aligned} \mathbf{E}\Big[\big(Z_1(x_j)Z_1(x'_j)\big)^2\Big] \;&=\; \mathbf{E}\left[\Big(2\cos\big(\langle W_1, x_j\rangle + U_1\big)\cos\big(\langle W_1, x_{j'}\rangle + U_1\big)\Big)^2\right] \\ &= \mathbf{E}\left[\Big(\cos\big(\langle W_1, x_j - x_{j'}\rangle\big) + \cos\big(\langle W_1, x_j + x_{j'}\rangle + 2U_1\big)\Big)^2\right] \\ &= \mathrm{I} + \mathrm{II} + \mathrm{III}, \end{aligned} \tag{A.11}$$

where we let

$$\begin{aligned} \mathrm{I} \;&=\; \mathbf{E}\Big[\cos\big(\langle W_1, x_j - x_{j'}\rangle\big)^2\Big] \\ \mathrm{II} \;&=\; \mathbf{E}\Big[2\cos\big(\langle W_1, x_j - x_{j'}\rangle\big)\cos\big(\langle W_1, x_j + x_{j'}\rangle + 2U_1\big)\Big] \\ \mathrm{III} \;&=\; \mathbf{E}\Big[\cos\big(\langle W_1, x_j + x_{j'}\rangle + 2U_1\big)^2\Big]. \end{aligned}$$

For the term I, we apply Jensen's inequality, followed by the formula (1.1) from Bochner's Theorem to obtain

$$\begin{aligned} \mathrm{I} \;&\geq\; \mathbf{E}\left[\cos(\langle W_1, x_j - x_{j'}\rangle)\right]^2 \\ &=\; k(x_j, x_{j'})^2. \end{aligned} \tag{A.12}$$

Next, the term II turns out to vanish. This is because we can apply the sum-of-angles identity again to obtain

$$\begin{aligned} \mathrm{II} \;&=\; \mathbf{E}\left[\cos(\langle W_1, -2x_{j'}\rangle - 2U_1)\right] \;+\; \mathbf{E}\left[\cos(\langle W_1, 2x_j\rangle + 2U_1)\right] \\ &=\; 0, \end{aligned} \tag{A.13}$$

where the last step uses the facts that W_1 and U_1 are independent and that for any fixed $r \in \mathbb{R}$, we have

$$\mathbf{E}[\cos(r \pm 2U_1)] \;=\; \mathrm{Re}\Big(\tfrac{e^{\sqrt{-1}r}}{2\pi}\int_0^{2\pi} e^{\pm\sqrt{-1}(2u)}du\Big) \;=\; 0.$$

Lastly, for the term III, we apply the sum-of-angles formula with $a = b$ to get

$$\begin{aligned} \mathrm{III} \;&=\; \tfrac{1}{2} \;+\; \tfrac{1}{2}\,\mathbf{E}\Big[\cos\big(\langle W_1, 2(x_j + x_{j'})\rangle + 4U_1\big)\Big] \\ &=\; \tfrac{1}{2}, \end{aligned} \tag{A.14}$$

where the expectation on the right vanishes due to the same reasoning that was used in (A.13). Altogether, we see that $\mathrm{I} + \mathrm{II} + \mathrm{III} \geq 1/2 + k(x_j, x_{j'})^2$, and combining this with equations (A.10) and (A.11) gives the lower bound

$$\mathrm{var}(\mathsf{Y}_{jj'}(1)) \;\geq\; \tfrac{1}{2}. \tag{A.15}$$

Hence, the condition (i) is satisfied with $c_1 = 1/2$, which completes the proof. □

Concluding the proof of Theorem 1. Combining Lemmas A.1 and A.2 with the triangle inequality shows there is an absolute constant $c > 0$ such that the bound

$$\sup_{t\in\mathbb{R}} \big|\tilde{F}_{s,N}(t) - F_s(t)\big| \;\leq\; \tfrac{\sqrt{\log(N)}}{\sqrt{N}} + \Big(\tfrac{c\log(2sn)^5}{s}\Big)^{1/4} \tag{A.16}$$

holds with probability at least $1 - c(1/s + \sqrt{\log(sn)^3/s} + 1/N)$. Due to this uniform approximation, classical arguments can be used to show that the quantiles of $\tilde{F}_{s,N}$ and F_s behave similarly, implying that the event $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\infty \leq \tilde{\varepsilon}_{1-\alpha}$ holds with probability close to $1 - \alpha$. For example, the arguments in the proof of Theorem 2.5 in Chernozhuokov et al. (2022) or the proof of Lemma 10.4 in Lopes (2022) can be used to show that (A.16) implies

$$\Big|\mathbf{P}\Big(\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\infty \leq \tilde{\varepsilon}_{1-\alpha}\Big) \;-\; (1-\alpha)\Big| \;\leq\; \tfrac{c\sqrt{\log(N)}}{\sqrt{N}} + \Big(\tfrac{c\log(2sn)^5}{s}\Big)^{1/4} \tag{A.17}$$

for some absolute constant $c > 0$. Finally, as $n \to \infty$, the assumptions of Theorem 1 ensure that the terms on the right side of (A.17) approach 0, which completes the proof. □

B Error estimation for RFF in hypothesis testing

This section looks at using $\tilde{\delta}_{1-\alpha}$ to estimate the error arising from RFF in the context of kernel-based hypothesis testing.

MMD statistic. Let $\mathcal{D}_x = \{x_1, \ldots, x_n\}$ and $\mathcal{D}_y = \{y_1, \ldots, y_n\}$ denote two datasets in $\mathbb{R}^d$, and consider the problem of testing the null hypothesis that both $\mathcal{D}_x$ and $\mathcal{D}_y$ were drawn in an i.i.d. manner from the same distribution. A well-known approach for solving this problem is based on the notion of Maximum Mean Discrepancy (MMD), which is a statistical distance that can be formulated in terms of kernels (Gretton et al., 2012).

For a given kernel k, an MMD test statistic can be defined as

$$T \;=\; \frac{1}{n(n-1)}\sum_{i\neq i'}^{n} k(x_i, x_{i'}) - \frac{2}{n^2}\sum_{i,j=1}^{n} k(x_i, y_j) + \frac{1}{n(n-1)}\sum_{j\neq j'}^{n} k(y_j, y_{j'}), \tag{B.1}$$

which is referred to as MMD_u^2 in the paper (Gretton et al., 2012). Alternatively, we may view T as a functional of the kernel, say $T = \psi(k)$.

In order to compute an approximation to T via RFF, one may use a corresponding statistic defined as $\tilde{T} = \psi(\tilde{k})$ with the approximate kernel $\tilde{k}$. In particular, we have

$$\tilde{T} \;=\; \frac{1}{n(n-1)} \sum_{i \neq i'}^{n} \tilde{k}(x_i, x_{i'}) - \frac{2}{n^2} \sum_{i,j=1}^{n} \tilde{k}(x_i, y_j) + \frac{1}{n(n-1)} \sum_{j \neq j'}^{n} \tilde{k}(y_j, y_{j'}). \tag{B.2}$$

It is also worth noting that $\tilde{T}$ can be obtained in an equivalent but computationally more efficient way. For this purpose, let $\mathsf{z}(\cdot) = \frac{1}{\sqrt{s}}(Z_1(\cdot), \ldots, Z_s(\cdot))$, with the functions $Z_1(\cdot), \ldots, Z_s(\cdot)$ defined as in Section 1.1, and let

$$\bar{\mathsf{z}}_x = \frac{1}{n} \sum_{j=1}^{n} \mathsf{z}(x_j) \quad \text{and} \quad \bar{\mathsf{z}}_y = \frac{1}{n} \sum_{j=1}^{n} \mathsf{z}(y_j),$$

which are both vectors in $\mathbb{R}^s$. Then, the statistic $\tilde{T}$ is expressible as

$$\tilde{T} \;=\; \tfrac{n^2}{n^2-n}\Big(\|\bar{\mathsf{z}}_x\|_2^2 - \tfrac{1}{n^2} \textstyle\sum_{j=1}^{n} \|\mathsf{z}(x_j)\|_2^2 \Big) \;- 2\big\langle \bar{\mathsf{z}}_x, \bar{\mathsf{z}}_y \big\rangle \;+\; \tfrac{n^2}{n^2-n}\Big(\|\bar{\mathsf{z}}_y\|_2^2 - \tfrac{1}{n^2} \textstyle\sum_{j=1}^{n} \|\mathsf{z}(y_j)\|_2^2 \Big),$$

which has the advantage that it can be computed with a cost that is linear n, rather than quadratic in n (as in (B.2)).

To assess the error of the RFF approximation using the framework developed in Sections 1 and 2, we estimate the 90% and 99% quantiles $\delta_{0.9}$ and $\delta_{0.99}$ of the error variable $|\psi(\tilde{k}) - \psi(k)| = |\tilde{T} - T|$ using Algorithm 1.

Data examples. We constructed three different versions of the pair $(\mathcal{D}_x, \mathcal{D}_y)$. Each version was constructed so that $|\mathcal{D}_x| = |\mathcal{D}_y| = 25000$ and $d = 10$. The first version of $(\mathcal{D}_x, \mathcal{D}_y)$ was obtained by uniformly subsampling 25000 rows and 10 columns from the datasets *YearPredictionMSD* (MSD) and *Buzz in social media* (Buzz), and the second version of $(\mathcal{D}_x, \mathcal{D}_y)$ was obtained in the same way from the datasets *SGEMM GPU kernel performance* (GPU) and *Gas Turbine CO and NOx Emission* (Emission). (The four named datasets are available in the repository (Dua and Graff, 2017).) In addition, the third version of $(\mathcal{D}_x, \mathcal{D}_y)$ was constructed with synthetic data by sampling 25000 points from the two multivariate Gaussian distributions $N(0, \frac{1}{10} \cdot \mathsf{I}_{10})$, and $N(0, (\frac{1}{10} + \eta) \cdot \mathsf{I}_{10})$, where $\eta > 0$ was chosen small enough so that detecting a difference with T was relatively challenging. More specifically, we selected $\eta = .0933$ so that the p-value derived from T (as in Corollary 11 of (Gretton et al., 2012)) was nearly equal to 5%.

Design of experiments. Our experiments in this section were organized analogously to those in Section 5.2. In particular, for a grid of s values ranging from 30 to 600, we generated 300 realizations of the approximate kernel $\tilde{k}$, and we applied Algorithm 1 to each such realization with $N = 30$ bootstrap iterations. The results for these experiments are displayed in Figures B.1 and B.2, where the three colored curves for $(\delta_{0.9}, \tilde{\delta}_{0.9}, \tilde{\delta}_{0.9}^{\mathrm{EXT}})$ and $(\delta_{0.99}, \tilde{\delta}_{0.99}, \tilde{\delta}_{0.99}^{\mathrm{EXT}})$ have the same interpretations as the corresponding curves in Figure 4. In the current context, all the curves were multiplied by the relevant value of $1/T$, so that they can be viewed on a more natural scale. Also, the curves for the extrapolated estimates $\tilde{\delta}_{0.9}^{\mathrm{EXT}}$ and $\tilde{\delta}_{0.99}^{\mathrm{EXT}}$ are based on a starting point of $s_0 = 50$. Lastly, the experiments were performed with three different kernels: the Gaussian kernel $\exp(-\|x - x'\|_2^2/2)$, the Laplacian kernel $\exp(-\|x - x'\|_1/2)$, and the Cauchy kernel $\prod_{j=1}^{10} 1/(1 + \Delta_j^2/2)$ where $\Delta = x - x'$.

Discussion of results. Figure B.1 shows that the estimates $\tilde{\delta}_{1-\alpha}$ and $\tilde{\delta}_{1-\alpha}^{\mathrm{EXT}}$ agree well with $\delta_{1-\alpha}$ across different choices of kernels and datasets when $1 - \alpha = 90\%$. The same pattern also appears in Figure B.2 for the case when $1 - \alpha = 99\%$, which is especially encouraging because the choice of $1 - \alpha = 99\%$ makes the estimation problem more challenging. Furthermore, it is notable that the same inexpensive choice $s_0 = 50$ leads to high-quality extrapolations for both choices of α.

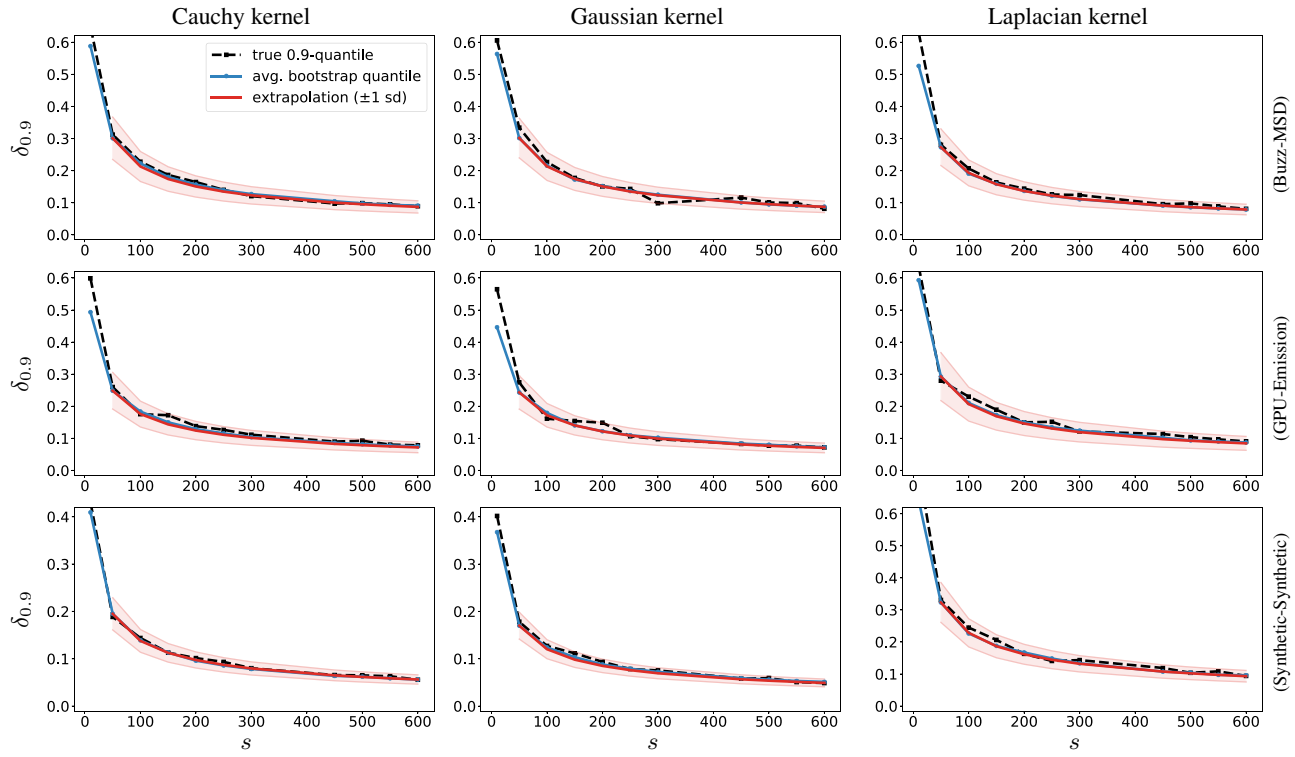

Figure B.1: (Estimation of $\delta_{0.9}$ for $|\psi(\tilde{k}) - \psi(k)| = |\tilde{T} - T|$.) The rows correspond to different pairs of datasets, and the columns correspond to different kernels.

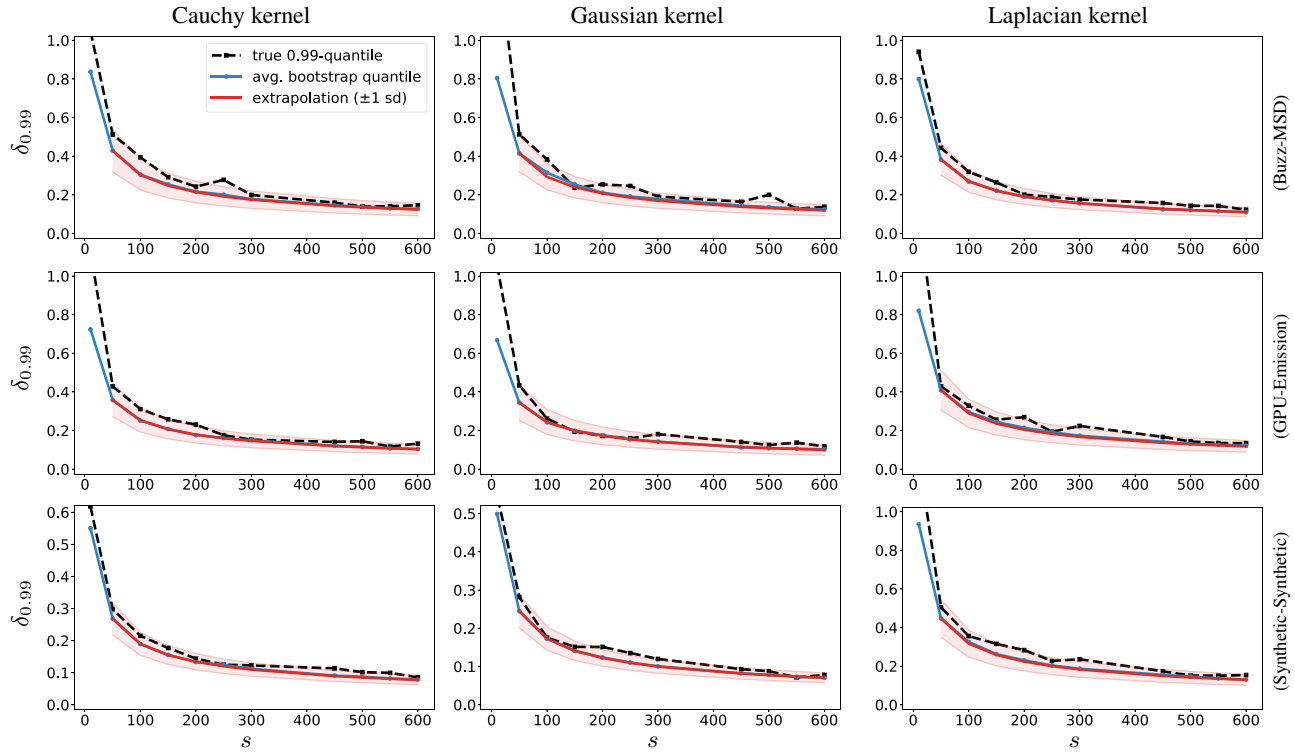

Figure B.2: (Estimation of $\delta_{0.99}$ for $|\psi(\tilde{k}) - \psi(k)| = |\tilde{T} - T|$.) The rows correspond to different pairs of datasets, and the columns correspond to different kernels.

C Additional results on error estimation for RFF in kernel matrix approximation

This appendix is a continuation of Section 5.1 from the main text, in which we present additional results for data that reside on the well-known 3-dimensional "Swiss roll" structure. Specifically, we used code provided by Marsland (2011) to generate $n = 20000$ data points. Apart from the choice of the dataset, the experiments here followed the same design and settings as in Section 5.1.

Figure C.1 displays the performance of $\tilde{\varepsilon}_{0.9}$ and $\tilde{\varepsilon}_{0.9}^{\mathrm{EXT}}$ in the task of estimating $\varepsilon_{0.9}$. The top and bottom rows of Figure C.1 correspond respectively to the cases when matrix approximation error is measured through the operator norm $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_{\mathrm{op}}$ and the ℓ_∞-norm $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\infty$. All the plots within Figure C.1 show that the estimates enjoy the same high degree of accuracy that was observed for the other datasets considered in Section 5.1 of the main text.

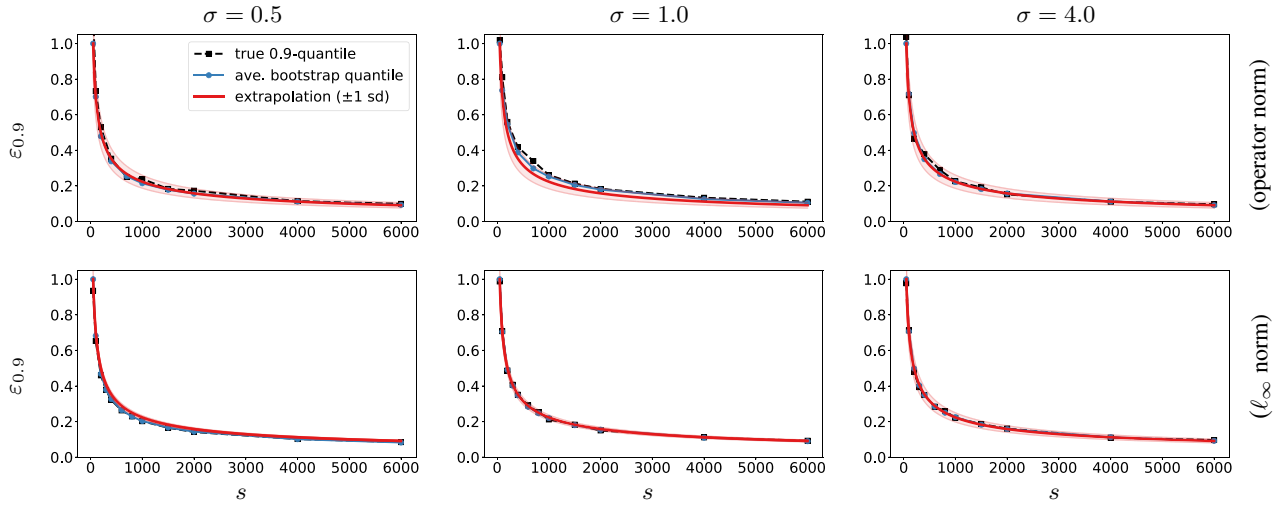

Figure C.1: (Estimation of $\varepsilon_{0.9}$ for $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_{\mathrm{op}}$ and $\|\widetilde{\mathsf{K}} - \mathsf{K}\|_\infty$). All plots are based on the Swiss roll dataset. The rows correspond to choices of matrix norm, and the columns correspond to choices of the kernel bandwidth σ.