Journal of Machine Learning Research 25 (2024) 1-75 Submitted 2/22; Revised 7/24; Published 8/24

Improved Random Features for Dot Product Kernels

Jonas Wacker JONAS.WACKER@GMAIL.COM
Motonobu Kanagawa MOTONOBU.KANAGAWA@EURECOM.FR
Data Science Department, EURECOM, France
Maurizio Filippone MAURIZIO.FILIPPONE@KAUST.EDU.SA
Statistics Program, KAUST, Saudi Arabia

Editor: Jean-Philippe Vert

Abstract

Dot product kernels, such as polynomial and exponential (softmax) kernels, are among the most widely used kernels in machine learning, as they enable modeling the interactions between input features, which is crucial in applications like computer vision, natural language processing, and recommender systems. We make several novel contributions for improving the efficiency of random feature approximations for dot product kernels, to make these kernels more useful in large scale learning. First, we present a generalization of existing random feature approximations for polynomial kernels, such as Rademacher and Gaussian sketches and TensorSRHT, using complex-valued random features. We show empirically that the use of complex features can significantly reduce the variances of these approximations. Second, we provide a theoretical analysis for understanding the factors affecting the efficiency of various random feature approximations, by deriving closed-form expressions for their variances. These variance formulas elucidate conditions under which certain approximations (e.g., TensorSRHT) achieve lower variances than others (e.g., Rademacher sketches), and conditions under which the use of complex features leads to lower variances than real features. Third, by using these variance formulas, which can be evaluated in practice, we develop a data-driven optimization approach to improve random feature approximations for general dot product kernels, which is also applicable to the Gaussian kernel. We describe the improvements brought by these contributions with extensive experiments on a variety of tasks and datasets.

Keywords: Random features, randomized sketches, dot product kernels, polynomial kernels, large scale learning

Contents

1. Introduction

Statistical learning methods based on *positive definite kernels*, namely Gaussian processes (Rasmussen and Williams, 2006) and kernel methods (Scholkopf and Smola, 2002), are among the most theoretically principled approaches in machine learning with competitive empirical performance. Due to their strong theoretical guarantees, these methods should be of primary choice in applications where the learning machine should behave in an anticipated manner, e.g., when high-stake decision-making is involved or when safety is required. However, a well-known drawback of these methods is their high computational costs, as naive implementations usually require the computational complexity of $\mathcal{O}(N^3)$ or at least $\mathcal{O}(N^2)$, where N is the training data size. This unfavorable scalability is an obstacle for these methods to handle a large amount of data. Moreover, it is problematic from a sustainability viewpoint, since these methods may perform essentially redundant computations and thus waste available computational resources.

The scalability issue has been a focus of research since the earliest literature (Wahba, 1990, Chapter 7), and many approximation methods for reducing the computational costs have been developed (e.g., Williams and Seeger 2000; Rahimi and Recht 2007; Titsias 2009; Hensman et al. 2018). One of the most successful approximations are those based on *random features*, initiated by Rahimi and Recht (2007). This approach constructs a random feature map Φ that transforms an input point $\boldsymbol{x}$ to a finite dimensional feature vector $\Phi(\boldsymbol{x}) \in \mathbb{R}^D$, so that the inner product of two feature maps $\Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{y})$ approximates the kernel value $k(\boldsymbol{x}, \boldsymbol{y})$ of the two input points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. The resulting computational complexity is dominated by the dimensionality D of the random features, and thus the computational costs can be drastically reduced if D is much smaller than the training data size N. Rahimi and Recht (2007) proposed *random Fourier features* for *shift-invariant kernels* on the Euclidean space $\mathbb{R}^d$. These are kernels that depend only on the difference of two input points, i.e., of the form $k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x} - \boldsymbol{y})$, such as the Gaussian and Matérn kernels. For a recent overview of random Fourier features and their extensions, see Liu et al. (2020).

Another important class of kernels are *dot product kernels*, which can be written as a function of the dot product (or the inner product) between two input points, i.e., kernels of the form $k(\boldsymbol{x}, \boldsymbol{y}) = k(\boldsymbol{x}^\top \boldsymbol{y})$. Representative examples include *polynomial kernels*, $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y} + \nu)^p$ with $\nu \geq 0$ and $p \in \mathbb{N}$, and *exponential kernels* (or *softmax kernels*), $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(\boldsymbol{x}^\top \boldsymbol{y} / \sigma^2)$ with $\sigma > 0$. These kernels can model the interactions between input features[1] (e.g., Agrawal et al., 2019), and thus are useful in applications such as genomic data analysis (Aschard, 2016; Weissbrod et al., 2016), recommender systems (Rendle, 2010; Blondel et al., 2016), computer vision (Lin et al., 2015; Gao et al., 2016; Fukui et al., 2016), and natural language processing (Yamada and Matsumoto, 2003; Chang et al., 2010; Vaswani et al., 2017). Recent notable applications of dot product kernels include *bilinear pooling* in computer vision (Lin et al., 2015), which essentially uses a polynomial kernel, and the *dot product attention mechanism* in the Transformer architecture (Vaswani et al., 2017), which uses an exponential kernel.

As for kernel methods in general, approximations are necessary to make use of dot product kernels in large scale learning. However, since dot product kernels are *not* shift-invariant, one cannot apply random Fourier features for their approximations, except for some specific cases (c.f. Pennington et al., 2015; Choromanski et al., 2021). Therefore, alternative random feature approximations have been proposed for dot product kernels in the literature, with a particular focus on polynomial kernels (Kar and Karnick, 2012; Pham and Pagh, 2013; Hamid et al., 2014; Avron et al., 2014; Ahle et al., 2020; Song et al., 2021). These approximations are based on *sketching* (Woodruff, 2014), which is a randomized linear projection of input feature vectors into a low dimensional space. Random feature approximations play a key role in the above applications of dot product kernels (e.g., Gao et al., 2016; Fukui et al., 2016; Choromanski et al., 2021).

This paper contributes to the above line of research, by suggesting various approaches to improving the efficiency of random feature approximations for dot product kernels. Our overarching goal is to make these kernels more useful in large scale learning, thereby widening their applicability. Specifically, we make the following contributions: (From now on, we refer to sketching-based random feature approximations for polynomial kernels as *polynomial sketches* for brevity.)

Complex-valued features. We propose a generalization of polynomial sketches using *complex-valued* random features. This generalization is applicable to all polynomial sketches[2], including those using i.i.d. Gaussian or Rademacher features (Kar and Karnick, 2012; Hamid et al., 2014) and structured sketches such as TensorSRHT (Hamid et al., 2014; Ahle et al., 2020). Our approach is an extension of complex-valued random features for the *linear* kernel discussed in Choromanski et al. (2017) to *polynomial* kernels. We empirically show that the generalized polynomial sketches using complex features are statistically more efficient than those using real features (in terms of the resulting variances) in particular for higher degree polynomial kernels, and that the former leads to better performance in downstream learning tasks. To corroborate these empirical findings, we provide a theoretical analysis of the variances of these sketches, as explained below.

[1] As explained in Section 5.1, any dot product kernel can be written as a weighted sum of polynomial kernels. Since each polynomial kernel models multiplicative interactions between input features (See Section 3), the resulting dot product kernel also implicitly models such interactions.

[2] There exists a recent line of research on improving the efficiency of polynomial sketches using a hierarchical feature construction (e.g., Ahle et al., 2020; Song et al., 2021). One can also use the proposed complex polynomial sketches as base sketches in such a hierarchical construction. Therefore, our contributions are complementary to this line of research.

Variance formulas. We derive *closed-form* formulas for the variances of the Gaussian and Rademacher polynomial sketches (Kar and Karnick, 2012; Hamid et al., 2014) and TensorSRHT (Hamid et al., 2014; Ahle et al., 2020) as well as for their complex generalizations. These variance formulas provide new insights into the factors affecting the efficiency of these polynomial sketches, complementing existing theoretical results (c.f. Kar and Karnick, 2012; Hamid et al., 2014; Ahle et al., 2020). Specifically, they elucidate conditions under which certain approximations (e.g., TensorSRHT) achieve lower variances than others (e.g, Rademacher sketches), and conditions under which the use of complex features leads to lower variances than real features. Importantly, these variance formulas can be evaluated in practice, and thus can be optimized; this is how we develop a novel optimization approach to random feature construction, explained next.

Optimized Maclaurin approximation for general dot product kernels. Using the derived variance formulas, we develop a data-driven optimization approach to random feature approximations for general dot product kernels, which is also applicable to the Gaussian kernel. Inspired from the randomized Maclaurin approximation of Kar and Karnick (2012), we use a finite-degree Maclaurin approximation of the kernel, given as a weighted sum of polynomial kernels of different degrees. Our approach optimizes the cardinalities of random features for approximating the polynomial kernels of different degrees (given a total number of random features), so as to minimize the mean square error of the approximate kernel with respect to the data distribution – we utilize the variance formulas to define this optimization objective. This optimized Maclaurin approach is compatible with exiting polynomial sketches as well as their complex generalizations, and enhances the efficiency of these sketches to achieve state-of-the-art performance, as we show in our experiments.

Extensive empirical comparison. We conduct extensive experiments to study the effectiveness of the suggested approaches. Our investigations include the approximations of polynomial and Gaussian kernels, and cover various random feature approximations. We study not only the quality of kernel approximation, but also the performance in downstream learning tasks of Gaussian process regression and classification. We generally observe that the proposed approaches lead to significant reduction of kernel approximation errors, and also to state-of-the-art performance in the downstream tasks on most datasets.

Software package. We provide a GitHub repository[3] with modern implementations for all the methods studied in this work supporting GPU acceleration and automatic differentiation in PyTorch (Paszke et al., 2019). Since version 1.8, PyTorch natively supports numerous linear algebra operations on complex numbers[4]. The same is true for NumPy (Harris et al., 2020) and TensorFlow (Abadi et al., 2016). Therefore, it is straightforward to implement the complex-valued polynomial sketches proposed in this work.

This paper is organized as follows. Section 2 presents preliminaries. In Section 3, we review polynomial sketches using i.i.d. random features and introduce their complex generalizations. We also provide a theoretical analysis and derive variance formulas. In Section 4, we study structured polynomial sketches and their complex generalizations, also deriving

[3] Our code is available at: https://github.com/joneswack/dp-rfs

[4] The PyTorch 1.8 release notes are available at: https://github.com/pytorch/pytorch/releases/tag/v1.8.0

their variance formulas. In Section 5, we study the approximation of general dot product kernels, and present the optimized Maclaurin approach. In Section 6, we report the results of extensive experiments. The appendix contains many supplementary materials, including proofs for theoretical results, additional experiments, and an explanation of Gaussian process regression and classification using complex random features.

2. Preliminaries

This section serves as preliminaries for describing our main contributions. We first introduce basic notation and definitions in Section 2.1. We then define positive definite kernels in Section 2.2.

2.1 Notation

Let $\mathbb{N}$ and $\mathbb{R}$ denote the sets of natural and real numbers, respectively, and let $\mathbb{R}^d$ denote the real vector space of dimension $d \in \mathbb{N}$. Let $\mathbb{C}$ be the set of complex numbers, and $\overline{c}$ for $c \in \mathbb{C}$ be the complex conjugate of c. Let $\mathrm{i} := \sqrt{-1}$ be the imaginary unit.

We use $\mathcal{X}$ to denote a set of input points, and we generally assume $\mathcal{X} \subseteq \mathbb{R}^d$. We write the vector-valued inputs by bold-faced letters, e.g., $\boldsymbol{x} \in \mathbb{R}^d$. For $\boldsymbol{x} := (x_1, \ldots, x_d)^\top \in \mathbb{R}^d$, let $\|\boldsymbol{x}\| := \|\boldsymbol{x}\|_2 := \sqrt{\sum_{i=1}^d x_i^2}$ be the 2-norm, and $\|\boldsymbol{x}\|_1 := \sum_{i=1}^d |x_i|$ be the 1-norm. We may interchangeably use $\|\boldsymbol{x}\|$ and $\|\boldsymbol{x}\|_2$ depending on the context.

For any two vectors $\boldsymbol{a} \in \mathbb{R}^{d_1}$ and $\boldsymbol{b} \in \mathbb{R}^{d_2}$, $\boldsymbol{a} \otimes \boldsymbol{b} := \mathrm{vec}(\boldsymbol{a}\boldsymbol{b}^\top) \in \mathbb{R}^{d_1 \cdot d_2}$ denotes the vectorized outer product between $\boldsymbol{a}$ and $\boldsymbol{b}$.

We denote by $\mathbb{E}[\cdot]$ the expected value and by $\mathbb{V}[\cdot]$ the variance of a random variable. For complex-valued vectors $\boldsymbol{z} = \boldsymbol{x} + \mathrm{i}\boldsymbol{y} \in \mathbb{C}^d$, with $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, we define $\mathcal{R}\{\boldsymbol{z}\} := \boldsymbol{x}$ and $\mathcal{I}\{\boldsymbol{z}\} := \boldsymbol{y}$ to be their real and imaginary parts, respectively.

We further define $\lfloor \cdot \rfloor$ and $\lceil \cdot \rceil$ to be the floor and ceil operators that round a floating point number down/up to the next integer, whereas $\mathrm{mod}(a, b)$ with $a, b \in \mathbb{N}$ is the arithmetic modulus that gives the rest after dividing a by b.

2.2 Positive Definite Kernels

Let $\mathcal{X}$ be a nonempty set. A symmetric function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is called *positive definite kernel*, if for every $m \in \mathbb{N}$, $x_1, \ldots, x_m \in \mathcal{X}$ and $c_1, \ldots, c_m \in \mathbb{R}$

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

We may simply call such k *kernel*. Popular examples include *Gaussian kernels* and *polynomial kernels*, among many others.

For any kernel k, there exists a corresponding *reproducing kernel Hilbert space (RKHS)* consisting of functions on $\mathcal{X}$. In kernel methods (Scholkopf and Smola, 2002), the RKHS provides implicit feature representations for points in $\mathcal{X}$, where each feature vector can be potentially infinite dimensional. A learning method is defined conceptually on such feature representations in the RKHS, but the resulting concrete algorithm can be formulated as a finite dimensional optimization problem defined through the evaluations of the kernel k

evaluated at given data points $x_1, \ldots, x_N$:

$$k(x_i, x_j), \quad i, j = 1, \ldots, N. \tag{1}$$

This reduction to a finite dimensional optimization problem is the core idea of kernel methods, enabled by the so-called kernel trick. However, the exact solution to the optimization problem often requires the computational complexity of $\mathcal{O}(N^3)$ or at least $\mathcal{O}(N^2)$ with N being the data size, which poses a computational challenge to kernel methods.

Similarly, any positive definite kernel k can be used to define a Gaussian process whose covariance function is k (Rasmussen and Williams, 2006). In Bayesian nonparametric learning, a Gaussian process is used to define a prior distribution over functions on $\mathcal{X}$, and the resulting posterior distribution is given in terms of the values of k evaluated on the data points. Like kernel methods, Gaussian processes also face a computational challenge, as naively computing the posterior requires the complexity of $\mathcal{O}(N^3)$ or at least $\mathcal{O}(N^2)$.

Various techniques have been proposed to speed up Gaussian processes and kernel methods by approximately computing the solution of interest. Approximations based on *random features* are one of the most successful approximation approaches, and these are the main topic of this paper.

3. Polynomial Sketches

We study here random feature approximations of *polynomial kernels*, defined as

$$k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y} + \nu)^p, \tag{2}$$

where $\nu \geq 0$ and $p \in \mathbb{N}$. We call such random features *polynomial sketches.* Since polynomial kernels are not shift-invariant, widely known random Fourier features (Rahimi and Recht, 2007) cannot be applied directly. Polynomial sketches are a fundamentally different approach, and can be understood as implicit randomized projections of the explicit high dimensional feature maps of polynomial kernels.

For simplicity, we focus on *homogeneous* polynomial kernels of the form

$$k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y})^p, \tag{3}$$

i.e., $\nu = 0$ in (2). The inhomogeneous case $\nu > 0$ can be reduced to the homogeneous case, by appending $\sqrt{\nu}$ to the input vectors, i.e., by setting $\tilde{\boldsymbol{x}} := [\boldsymbol{x}^\top, \sqrt{\nu}]^\top \in \mathbb{R}^{d+1}$ and $\tilde{\boldsymbol{y}} := [\boldsymbol{y}^\top, \sqrt{\nu}]^\top \in \mathbb{R}^{d+1}$, we have

$$(\boldsymbol{x}^\top \boldsymbol{y} + \nu)^p = (\tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{y}})^p$$

In this way, polynomial sketches for the homogeneous case can also be applied to the inhomogeneous case.

We first review existing polynomial sketches with i.i.d. real-valued features in Section 3.1. In Section 3.2, we propose polynomial sketches with *complex-valued features.* These complex-valued sketches are an extension of the complex-valued sketches for the *linear* kernel discussed in Choromanski et al. (2017) to *polynomial* kernels. We derive the variance formulas of these complex sketches in Section 3.3 and present a probabilistic error bound in Section 3.4.

3.1 Real-valued Polynomial Sketches

We first study polynomial sketches proposed by Kar and Karnick (2012), which are also discussed in Hamid et al. (2014). We do not cover here TensorSketch of Pham and Pagh (2013), as it is conceptually different from the other polynomial sketches discussed in this paper.[5]

Let $D \in \mathbb{N}$ be the number of random features, and $p \in \mathbb{N}$ be the degree of the polynomial kernel (3). Suppose we generate $p \times D$ i.i.d. random vectors

$$\boldsymbol{w}_{i,\ell} \in \mathbb{R}^d \quad \text{satisfying} \quad \mathbb{E}[\boldsymbol{w}_{i,\ell}\boldsymbol{w}_{i,\ell}^\top] = \boldsymbol{I}_d, \quad i \in \{1, \dots, p\}, \quad \ell \in \{1, \dots, D\}, \tag{4}$$

where $\boldsymbol{I}_d \in \mathbb{R}^{d \times d}$ denotes the identity matrix.

Then we define a random feature map as

$$\Phi_{\mathcal{R}}(\boldsymbol{x}) := \frac{1}{\sqrt{D}} \left[(\prod_{i=1}^{p} \boldsymbol{w}_{i,1}^\top \boldsymbol{x}), \dots, (\prod_{i=1}^{p} \boldsymbol{w}_{i,D}^\top \boldsymbol{x}) \right]^\top \in \mathbb{R}^D. \tag{5}$$

The resulting approximation of the polynomial kernel (3) is given by

$$\hat{k}_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y}) := \Phi_{\mathcal{R}}(\boldsymbol{x})^\top \Phi_{\mathcal{R}}(\boldsymbol{y}), \tag{6}$$

which is unbiased, as the expectation with respect to the random vectors (4) gives

$$\mathbb{E}\left[\Phi_{\mathcal{R}}(\boldsymbol{x})^\top \Phi_{\mathcal{R}}(\boldsymbol{y})\right] = \frac{1}{D} \sum_{\ell=1}^{D} \prod_{i=1}^{p} \boldsymbol{x}^\top \mathbb{E}[\boldsymbol{w}_{i,\ell}\boldsymbol{w}_{i,\ell}^\top] \boldsymbol{y} = (\boldsymbol{x}^\top \boldsymbol{y})^p.$$

Kar and Karnick (2012) suggest to define random vectors in (4) using the Rademacher distribution: each element of $\boldsymbol{w}_{i,\ell}$ is independently drawn from $\{-1, 1\}$ with equal probability. We study later how the distribution of the random vectors affects the quality of kernel approximation.

Implicit sketching of high-dimensional features. The random feature map (5) can be interpreted as a linear sketch (projection) of an explicit high-dimensional feature vector for the polynomial kernel. To describe this, consider the case $D = 1$ and let $\boldsymbol{w}_i = (w_{i,1}, \dots, w_{i,d})^\top \in \mathbb{R}^d$, $i = 1, \dots, p$, be i.i.d. random vectors satisfying $\mathbb{E}[\boldsymbol{w}_i \boldsymbol{w}_i^\top] = \boldsymbol{I}_d$. Then, the random feature map $\Phi_{\mathcal{R}}(\boldsymbol{x})$ (which is one dimensional in this case) for $\boldsymbol{x} := (x_1, \dots, x_d)^\top$ is given by

$$\Phi_{\mathcal{R}}(\boldsymbol{x}) = \prod_{i=1}^{p} \boldsymbol{w}_i^\top \boldsymbol{x} = \prod_{i=1}^{p} \sum_{j=1}^{d} w_{i,j} x_j = \sum_{j_1=1,\dots,j_p=1}^{d} w_{1,j_1} x_{j_1} \cdots w_{p,j_p} x_{j_p} = \boldsymbol{w}^{(p)\top} \boldsymbol{x}^{(p)}, \tag{7}$$

where $\boldsymbol{x}^{(p)} \in \mathbb{R}^{d^p}$ and $\boldsymbol{w}^{(p)} \in \mathbb{R}^{d^p}$ are defined as (recall the notation in Section 2.1)

$$\boldsymbol{x}^{(p)} := \boldsymbol{x} \underbrace{\otimes \cdots \otimes}_{p \,\text{times}} \boldsymbol{x} \in \mathbb{R}^{d^p}, \quad \boldsymbol{w}^{(p)} := \boldsymbol{w} \underbrace{\otimes \cdots \otimes}_{p \,\text{times}} \boldsymbol{w} \in \mathbb{R}^{d^p},$$

[5] However, we will include TensorSketch in our empirical evaluation in Section 6.

Therefore, for $D = 1$, the approximate kernel is given as

$$\hat{k}_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y}) := \Phi_{\mathcal{R}}(\boldsymbol{x}) \cdot \Phi_{\mathcal{R}}(\boldsymbol{y}) = \boldsymbol{w}^{(p)\top}\boldsymbol{x}^{(p)} \cdot \boldsymbol{w}^{(p)\top}\boldsymbol{y}^{(p)} \tag{8}$$

On the other hand, the polynomial kernel can be written as (Scholkopf and Smola, 2002, Proposition 2.1):

$$(\boldsymbol{x}^\top \boldsymbol{y})^p = (\boldsymbol{x}^{(p)})^\top \boldsymbol{y}^{(p)},$$

where $\boldsymbol{y}^{(p)} = \boldsymbol{y} \otimes \cdots \otimes \boldsymbol{y} \in \mathbb{R}^{d^p}$. Thus, $\boldsymbol{x}^{(p)}$ and $\boldsymbol{y}^{(p)}$ are the exact feature maps of the input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$, respectively. The comparison of this expression with (8) implies that the random feature map $\Phi_{\mathcal{R}}(\boldsymbol{x}) = \boldsymbol{w}^{(p)\top}\boldsymbol{x}^{(p)} \in \mathbb{R}$ in (7) is a projection of the exact feature map $\boldsymbol{x}^{(p)} \in \mathbb{R}^{d^p}$ onto $\mathbb{R}$.

Similarly, if $D > 1$, the random feature map $\Phi_{\mathcal{R}}(\boldsymbol{x}) \in \mathbb{R}^D$ in (5) can be interpreted as a projection of the exact feature map $\boldsymbol{x}^{(p)}$ onto $\mathbb{R}^D$. A remarkable point of this random feature map is that it can be obtained without constructing the exact feature vector $\boldsymbol{x}^{(p)}$, the latter being infeasible if d or p is large.[6] Indeed, the computational complexity of constructing the random feature map $\Phi_{\mathcal{R}}(\boldsymbol{x})$ is $\mathcal{O}(pdD)$, while the exact feature map $\boldsymbol{x}^{(p)}$ requires $\mathcal{O}(d^p)$.

3.2 Complex-valued Polynomial Sketches

We now introduce complex-valued polynomial sketches, one of our novel contributions. We do this by extending the analysis of Choromanski et al. (2017) for linear sketches to polynomial sketches.[7]

As before, without loss of generality, we focus on approximating the homogeneous polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y})^p$ of degree $p \in \mathbb{N}$. Let $D \in \mathbb{N}$. Suppose we generate $p \times D$ *complex-valued* random vectors satisfying

$$\boldsymbol{z}_{i,j} \in \mathbb{C}^d \quad \text{satisfying} \quad \mathbb{E}[\boldsymbol{z}_{i,j}\overline{\boldsymbol{z}_{i,j}}^\top] = \boldsymbol{I}_d, \quad i \in \{1, \dots, p\}, \quad j \in \{1, \dots, D\} \tag{9}$$

We then define a *complex-valued random feature map* as

$$\Phi_{\mathcal{C}}(\boldsymbol{x}) := \frac{1}{\sqrt{D}} \left[(\prod_{i=1}^{p} \boldsymbol{z}_{i,1}^\top \boldsymbol{x}), \dots, (\prod_{i=1}^{p} \boldsymbol{z}_{i,D}^\top \boldsymbol{x}) \right]^\top \in \mathbb{C}^D, \quad \boldsymbol{x} \in \mathbb{R}^d, \tag{10}$$

and the resulting approximate kernel as

$$\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y}) := \Phi_{\mathcal{C}}(\boldsymbol{x})^\top \overline{\Phi_{\mathcal{C}}(\boldsymbol{y})} = \frac{1}{D} \sum_{j=1}^{D} \prod_{i=1}^{p} (\boldsymbol{z}_{i,j}^\top \boldsymbol{x}) \overline{(\boldsymbol{z}_{i,j}^\top \boldsymbol{y})}, \quad \boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d. \tag{11}$$

[6] The exact feature expansion $\boldsymbol{x}^{(p)}$ leads to d^p dimensional vectors. By grouping up equal terms, we can reduce the dimensionality to $\binom{d+p-1}{p}$, which still leads to unrealistic dimensional feature vectors as soon as d is large. For example, working with MNIST images of size 28x28 ($d = 784$) leads to 307,720 features for $p = 2$ and to 80,622,640 features for $p = 3$. This justifies the need for randomized approximations of the polynomial kernel.

[7] More specifically, Choromanski et al. (2017) analyze the variance of the *real part* of the approximate complex-valued kernel in Eq. (11) for $p = 1$. In contrast, we study Eq. (11) with generic $p \in \mathbb{N}$, and analyze the variance of Eq. (11) itself, including both the real and imaginary parts.

Eq. (11) is a generalization of the approximate kernel (6) with real-valued features, as Eq. (6) can be recovered by defining the complex random vectors $\boldsymbol{z}_{i,k}$ in Eq. (9) as real random vectors $\boldsymbol{w}_{i,k}$ in Eq. (4); in this case the requirement $\mathbb{E}[\boldsymbol{z}_{i,j}\overline{\boldsymbol{z}_{i,j}}^\top] = \mathbb{E}[\boldsymbol{w}_{i,j}\boldsymbol{w}_{i,j}^\top] = \boldsymbol{I}_d$ is satisfied.

For example, complex-valued random vectors $\boldsymbol{z}_{i,j}$ satisfying Eq. (9) can be generated as follows.

Example 1 *Suppose we generate $2 \times p \times D$ independent real-valued random vectors*

$$\boldsymbol{v}_{i,j},\ \boldsymbol{w}_{i,j} \in \mathbb{R}^d \quad \textit{satisfying} \quad \mathbb{E}[\boldsymbol{v}_{i,j}] = \mathbb{E}[\boldsymbol{w}_{i,j}] = \boldsymbol{0}, \quad \mathbb{E}[\boldsymbol{v}_{i,j}\boldsymbol{v}_{i,j}^\top] = \mathbb{E}[\boldsymbol{w}_{i,j}\boldsymbol{w}_{i,j}^\top] = \boldsymbol{I}_d \tag{12}$$

for $i \in \{1, \dots, p\}$, $j \in \{1, \dots, D\}$. Then one can define complex-valued random vectors (9) *as*

$$\boldsymbol{z}_{i,j} := \sqrt{\frac{1}{2}}(\boldsymbol{v}_{i,j} + \mathrm{i}\boldsymbol{w}_{i,j}) \in \mathbb{C}^d, \quad i \in \{1, \dots, p\}, \quad j \in \{1, \dots, D\}. \tag{13}$$

The following two examples are specific cases of Example 1 and are complex versions of the real-valued Rademacher and Gaussian sketches discussed previously.

Example 2 (Complex Rademacher Sketch) *In Example 1, suppose that elements of random vectors $\boldsymbol{v}_{i,j}$ and $\boldsymbol{w}_{i,j}$ are independently sampled from the Rademacher distribution, i.e., sampled uniformly from $\{1, -1\}$. Then the resulting random vectors $\boldsymbol{v}_{i,j}$, $\boldsymbol{w}_{i,j}$ satisfy the conditions in Eq.* (12) *and thus the complex random vectors in Eq.* (13) *satisfy the condition Eq.* (9)*.*

Example 3 (Complex Gaussian Sketch) *In Example 1, suppose that elements of random vectors $\boldsymbol{v}_{i,j}$ and $\boldsymbol{w}_{i,j}$ are independently sampled from the standard Gaussian distribution, $\mathcal{N}(0, 1)$. Then the resulting random vectors $\boldsymbol{v}_{i,j}$, $\boldsymbol{w}_{i,j}$ satisfy the conditions in Eq.* (12) *and thus the complex random vectors in Eq.* (13) *satisfy the condition Eq.* (9)*.*

Example 4 *Suppose the elements of each random vector $\boldsymbol{z}_{i,j} \in \mathbb{C}^d$ are independently sampled from the uniform distribution on $\{1, -1, \mathrm{i}, -\mathrm{i}\}$. Then the requirement in Eq.* (9) *is satisfied.*

Example 4 is essentially identical to the complex Rademacher sketch in Example 2, in that each element of $\boldsymbol{z}_{i,j}$ in Example 4 can be obtained by multiplying $e^{\mathrm{i}\pi/4}$ to an element of $\boldsymbol{z}_{i,j}$ in Example 2, and vice versa. The multiplication by $e^{\mathrm{i}\pi/4}$ is equivalent to rotating an element counter-clockwise by 45 degrees. See Fig. 1 for an illustration. One can see that this multiplication by $e^{\mathrm{i}\pi/4}$ does not change the resulting approximate kernel (11). In this sense, the constructions of Example 2 and Example 4 are equivalent. However, the sketch in Example 4 gives a computational advantage over Example 2: Since every element of each random vector $\boldsymbol{z}_{i,j}$ is either real *or* imaginary, the inner products $\boldsymbol{z}_{i,j}^\top \boldsymbol{x}$ in Eq. (10) can be computed at the same cost as for real polynomial sketches.

We show in the following proposition that the approximate kernel (11) is an unbiased estimator of the polynomial kernel $(\boldsymbol{x}^\top \boldsymbol{y})^p$.

Proposition 1 *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ be arbitrary, and $\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y})$ be the approximate kernel in* (11)*. Then we have*

$$\mathbb{E}[\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y})] = (\boldsymbol{x}^\top \boldsymbol{y})^p$$

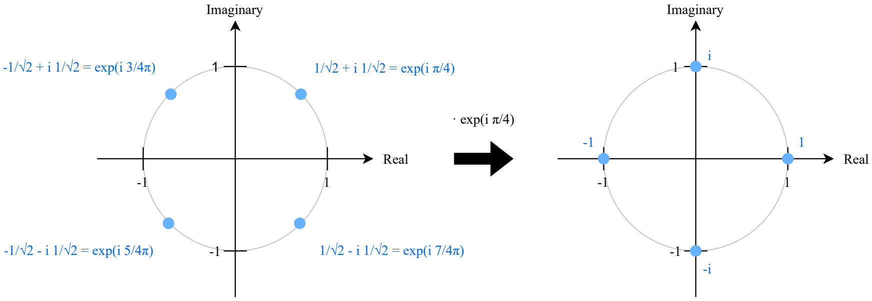
Imaginary
-1/√2 + i 1/√2 = exp(i 3/4π)
1/√2 + i 1/√2 = exp(i π/4)
1
-1
1
Real
-1/√2 - i 1/√2 = exp(i 5/4π)
-1
1/√2 - i 1/√2 = exp(i 7/4π)
· exp(i π/4)
Imaginary
i
1
-1
1
-1
1
Real
-1
-i

Figure 1: Multiplying each element of a random vector $\boldsymbol{z}_{i,j}$ in Example 2 by $\exp(\mathrm{i}\frac{\pi}{4})$ corresponds to a counter-clockwise rotation of that element by 45 degrees on the complex plane. The support of the resulting elements is $\{1, -1, \mathrm{i}, -\mathrm{i}\}$ and the construction of Example 4 is obtained.

Proof Since Eq. (11) is the empirical average of D terms, it is sufficient to show the unbiasedness of each term. To this end, we consider here the case $D = 1$ and drop the index j. We have

$$\mathbb{E}\left[\prod_{i=1}^{p}\left(\boldsymbol{z}_i^\top \boldsymbol{x}\right)\overline{\left(\boldsymbol{z}_i^\top \boldsymbol{y}\right)}\right] = \prod_{i=1}^{p}\mathbb{E}\left[\left(\boldsymbol{z}_i^\top \boldsymbol{x}\right)\overline{\left(\boldsymbol{z}_i^\top \boldsymbol{y}\right)}\right] = \prod_{i=1}^{p}\boldsymbol{x}^\top\mathbb{E}\left[\boldsymbol{z}_i\overline{\boldsymbol{z}_i}^\top\right]\boldsymbol{y} = (\boldsymbol{x}^\top\boldsymbol{y})^p.$$

where we used Eq. (9) in the last identity. ■

3.3 Variance of Complex-valued Polynomial Sketches

We now study the variance of the approximate kernel (11) with the complex-valued random feature map (10). We consider the case $D = 1$ and drop the index j:

$$\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{p}\left(\boldsymbol{z}_i^\top \boldsymbol{x}\right)\overline{\left(\boldsymbol{z}_i^\top \boldsymbol{y}\right)}. \tag{14}$$

The variance of the case $D > 1$ can be obtained by dividing the variance of Eq. (14) by D, since the approximate kernel (11) is the average of D i.i.d. copies of Eq. (14). We denote by $z_{i,k}$ the k-th element of $\boldsymbol{z}_i$.

Note that the variance of a complex random variable $Z \in \mathbb{C}$ is defined by

$$\mathbb{V}[Z] := \mathbb{E}[|Z - \mathbb{E}[Z]|^2] = \mathbb{E}[(Z - \mathbb{E}[Z])\overline{(Z - \mathbb{E}[Z])}] = \mathbb{E}[|Z|^2] - |\mathbb{E}[Z]|^2$$

Theorem 2 below characterizes the variance in terms of the input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ and the distribution of the complex weight vectors (9). The proof is given in Appendix A.1.

Theorem 2 *Let $\boldsymbol{x} := (x_1, \ldots, x_d)^\top \in \mathbb{R}^d$ and $\boldsymbol{y} := (y_1, \ldots, y_d)^\top \in \mathbb{R}^d$ be any input vectors. Let $\boldsymbol{z}_1, \ldots, \boldsymbol{z}_p \in \mathbb{C}^d$ be i.i.d. random vectors satisfying (9), such that elements $z_{i1}, \ldots, z_{id}$ of each vector $\boldsymbol{z}_i = (z_{i1}, \ldots, z_{id})^\top$ are themselves i.i.d. Let $\boldsymbol{z} = (z_1, \ldots, z_d)^\top \in \mathbb{C}^d$ be a random* *vector independently and identically distributed as $\boldsymbol{z}_1, \dots, \boldsymbol{z}_p$, and write $z_k = a_k + \mathrm{i}b_k$ with $a_k, b_k \in \mathbb{R}$. Suppose*

$$\mathbb{E}[a_k b_k] = 0, \quad \mathbb{E}[a_k^2] = q, \quad \mathbb{E}[b_k^2] = 1 - q \quad \textit{where} \quad 0 \leq q \leq 1. \tag{15}$$

Then, for the approximate kernel (14)*, we have*

$$\begin{aligned}\mathbb{V}[\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y})] = \Bigg(&\sum_{k=1}^{d} \mathbb{E}[|z_k|^4] x_k^2 y_k^2 + \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2 - \sum_{k=1}^{d} x_k^2 y_k^2 \\ &+ \left((2q-1)^2 + 1\right) \left((\boldsymbol{x}^\top \boldsymbol{y})^2 - \sum_{k=1}^{d} x_k^2 y_k^2\right)\Bigg)^p - (\boldsymbol{x}^\top \boldsymbol{y})^{2p} \end{aligned} \tag{16}$$

Theorem 2 applies to a spectrum of complex polynomial sketches in terms of q, where the case $q = 1$ is the case of real-valued polynomial sketches in Eq. (8). Indeed, to our knowledge, this result is also new for real-valued polynomial sketches. This variance formula is not only of theoretical interest, but also offers a way of estimating the variance from data. It will be used later to define the objective function of the proposed optimized Maclaurin approach.

Condition in Eq. (15). The key condition is Eq. (15),[8] where the constant q is the average length of the real part a_k of each random element $z_k = a_k + \mathrm{i}b_k$. Note that $\mathbb{E}[b_k^2] = 1 - q$ follows from $\mathbb{E}[a_k^2] = q$ since $1 = \mathbb{E}[|z_k|^2] = a_k^2 + b_k^2$. Eq. (15) is satisfied for Examples 2, 3 and 4 with $q = 1/2$ and for the real-valued Rademacher and Gaussian sketches with $q = 1$. If z_k is sampled uniformly from $\{\mathrm{i}, -\mathrm{i}\}$, which is eligible as it satisfies Eq. (9), then $q = 0$. If z_k is sampled uniformly from $\{1, -1\}$ with probability q and from $\{\mathrm{i}, -\mathrm{i}\}$ with probability $1 - q$, then Eq. (15) is satisfied with this q.

Lower Bound. The variance in Eq. (16) can be lower-bounded by using Jensen's inequality $\mathbb{E}[|z_k|^4] \geq (\mathbb{E}[|z_k|^2])^2 = 1$:

$$\mathbb{V}[\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y})] \geq \left(\|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2 + \left((2q-1)^2 + 1\right) \left((\boldsymbol{x}^\top \boldsymbol{y})^2 - \sum_{k=1}^{d} x_k^2 y_k^2\right)\right)^p - (\boldsymbol{x}^\top \boldsymbol{y})^{2p}. \tag{17}$$

Eq. (17) is the smallest possible variance attainable by complex polynomial sketches satisfying the conditions in Theorem 2. For $q = 1/2$, this lower bound is attained by the complex Rademacher sketch (Example 2) and its equivalent construction (Example 4), for which we have $\mathbb{E}[|z_k|^4] = 1$. On the other hand, for the complex Gaussian sketch (Example 3) we have $\mathbb{E}[|z_k|^4] = \mathbb{E}[(a_k^2 + b_k^2)^2] = 2$.

[8] Eq. (15) implies that z_k is a *proper* complex random variable (Neeser and Massey, 1993).

Concrete Examples. Below, we summarize the variance formula for the real and complex Rademacher sketches, and the real and complex Gaussian sketches:

$$\textbf{(Real Radem.)}\quad \mathbb{V}\left[\hat{k}_{\mathcal{R}}(\boldsymbol{x},\boldsymbol{y})\right] = \left(\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2 + 2\left[(\boldsymbol{x}^\top\boldsymbol{y})^2 - \sum_{k=1}^d x_k^2 y_k^2\right]\right)^p - (\boldsymbol{x}^\top\boldsymbol{y})^{2p}, \tag{18}$$

$$\textbf{(Comp. Radem.)}\quad \mathbb{V}[\hat{k}_{\mathcal{C}}(\boldsymbol{x},\boldsymbol{y})] = \left(\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2 + (\boldsymbol{x}^\top\boldsymbol{y})^2 - \sum_{k=1}^d x_k^2 y_k^2\right)^p - (\boldsymbol{x}^\top\boldsymbol{y})^{2p} \tag{19}$$

$$\textbf{(Real Gauss.)}\quad \mathbb{V}\left[\hat{k}_{\mathcal{R}}(\boldsymbol{x},\boldsymbol{y})\right] = \left(\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2 + 2(\boldsymbol{x}^\top\boldsymbol{y})^2\right)^p - (\boldsymbol{x}^\top\boldsymbol{y})^{2p}. \tag{20}$$

$$\textbf{(Comp. Gauss.)}\quad \mathbb{V}[\hat{k}_{\mathcal{C}}(\boldsymbol{x},\boldsymbol{y})] = \left(\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2 + (\boldsymbol{x}^\top\boldsymbol{y})^2\right)^p - (\boldsymbol{x}^\top\boldsymbol{y})^{2p} \tag{21}$$

Comparing the Real and Complex Polynomial Sketches. Let us now compare the variances of real ($q = 1$) and complex ($q \neq 1$) polynomial sketches. First, it is easy to see that the variance of the complex Gaussian polynomial sketch (Eq. (21)) is upper-bounded by the variance of the real Gaussian sketch (Eq. (20)). For the lower bound in Eq. (17), which is attained by the Rademacher sketches, a more detailed analysis is needed. To this end, consider the term that depends on q:

$$\left((2q-1)^2 + 1\right)\left((\boldsymbol{x}^\top\boldsymbol{y})^2 - \sum_{k=1}^d x_k^2 y_k^2\right)$$

The variance in Eq. (16) is a monotonically increasing function of this term. Suppose

$$(\boldsymbol{x}^\top\boldsymbol{y})^2 - \sum_{k=1}^d x_k^2 y_k^2 = \sum_{i=1}^d \sum_{\substack{j=1\\ j\neq i}}^d x_i x_j y_i y_j \geq 0 \tag{22}$$

Then $q = 1/2$ (e.g., complex sketches in Examples 2, 3 and 4) makes the term the smallest, while $q = 1$ and $q = 0$ (purely real and imaginary polynomial sketches) makes it the largest. In other words, for input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ satisfying Eq. (22), complex-valued sketches with $q = 1/2$ result in a lower variance than the real-valued counterparts with $q = 1$. On the other hand, if Eq. (22) does not hold, real-valued sketches result in a lower variance than the complex-valued counterparts.

Therefore, whether complex-valued Rademacher sketches ($q = 1/2$) yield a lower variance than real-valued Rademacher sketches ($q = 1$) depends on whether Eq. (22) holds. For example, Eq. (22) holds true if input vectors $\boldsymbol{x} = (x_1, \ldots, x_d)^\top$ and $\boldsymbol{y} = (y_1, \ldots, y_d)^\top$ are *nonnegative*: $x_1, ..., x_d \geq 0$ and $y_1, \ldots, y_d \geq 0$. Nonnegative input vectors are ubiquitous in real-world applications, e.g., where each input feature represents the amount of a certain quantity, where input vectors are given by bag-of-words representations, one-hot encoding (categorical data), or min-max feature scaling, and where they are outputs of a ReLU neural network[9]. For such applications with nonnegative input vectors, complex-valued polynomial sketches always yield a smaller variance than the real-valued counterparts.

[9] c.f. DeepFried Convnets (Yang et al., 2015) and fine-grained image recognition (Gao et al., 2016).

3.4 Probabilistic Error Bounds for Rademacher Sketches

We present here probabilistic error bounds for the approximate kernel in Eq. (11) in terms of the number D of random features, using the variance formula obtained in the previous subsection and focusing on Rademacher sketches. The proof of the following result is given in Appendix A.2.

Theorem 3 *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ be arbitrary input vectors. For $0 \leq q \leq 1$, consider a polynomial sketch in Theorem 2 such that $\mathbb{E}[|z_k|^4] = 1$ and thus it attains the variance in Eq. (17). Define a constant $\sigma^2 \geq 0$ by*

$$\sigma^2 := \frac{1}{\|\boldsymbol{x}\|^{2p}\|\boldsymbol{y}\|^{2p}} \left[\left(\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2 + \left((2q-1)^2+1\right)\left((\boldsymbol{x}^\top\boldsymbol{y})^2 - \sum_{k=1}^{d} x_k^2 y_k^2\right) \right)^p - (\boldsymbol{x}^\top\boldsymbol{y})^{2p} \right]$$

Let $\epsilon, \delta > 0$ be arbitrary, and $D \in \mathbb{N}$ be such that

$$D \geq 2\left(\frac{2}{3\epsilon} + \frac{\sigma^2}{\epsilon^2}\right)\log\left(\frac{2}{\delta}\right). \tag{23}$$

Then, for the approximate kernel $\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y})$ in Eq. (11), we have

$$\Pr\left[\left|\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y}) - (\boldsymbol{x}^\top\boldsymbol{y})^p\right| \leq \epsilon \|\boldsymbol{x}\|_1^p \|\boldsymbol{y}\|_1^p\right] \geq 1 - \delta.$$

Eq. (23) shows that the required number D of random features to achieve the relative accuracy of ε (where the "relative" is with respect to $\|\boldsymbol{x}\|_1^p \|\boldsymbol{y}\|_1^p$) with probability at least $1-\delta$. For small ϵ, the second term σ^2/ϵ^2 dominates the first term $2/(3\epsilon)$. This second term depends on σ^2, which is a scaled version of the variance in Eq. (17) of the approximate kernel for $D = 1$. Thus, if the variance in Eq. (17) is smaller (resp. larger), one needs a smaller (resp. larger) number of random features to achieve the relative accuracy of ϵ.

Let us now compare the real-valued ($q = 1$) and complex-valued ($q = 1/2$) Rademacher sketches. As discussed earlier, the complex Rademacher sketch has a smaller variance than the real Rademacher sketch when the inequality in Eq. (22) holds for the two input vectors $\boldsymbol{x}, \boldsymbol{y}$. In particular, this inequality always holds when the input vectors are nonnegative. Therefore, in this case, the complex Rademacher sketch requires a smaller number of random features than the real Rademacher sketch to achieve a given accuracy.

When the inequality in Eq. (22) holds, the advantage of the complex Rademacher sketch becomes more significant for larger p. We illustrate this behavior in Fig. 2, where vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ are sampled randomly in the positive quadrant by first drawing $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{y}}$ from $\mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$ and then by computing $\boldsymbol{x} = |\tilde{\boldsymbol{x}}|/\|\tilde{\boldsymbol{x}}\|, \boldsymbol{y} = |\tilde{\boldsymbol{y}}|/\|\tilde{\boldsymbol{y}}\|$. (Here $|\tilde{\boldsymbol{x}}|$ denotes the vector whose elements are the absolute values of the elements of $\tilde{\boldsymbol{x}}$.) In the figure we report the mean squared error of the approximate kernel for $d = 8$ and $d = 32$ for Gaussian and Rademacher sketches. To facilitate quantitative analysis of the improvement offered by complex random features, we also report the ratio between the mean squared error obtained by complex and real random features. As expected from the theoretical analysis, the results show a decreasing error ratio for increasing values of p, with an improvement of Rademacher over Gaussian, which is slightly larger for lower d.

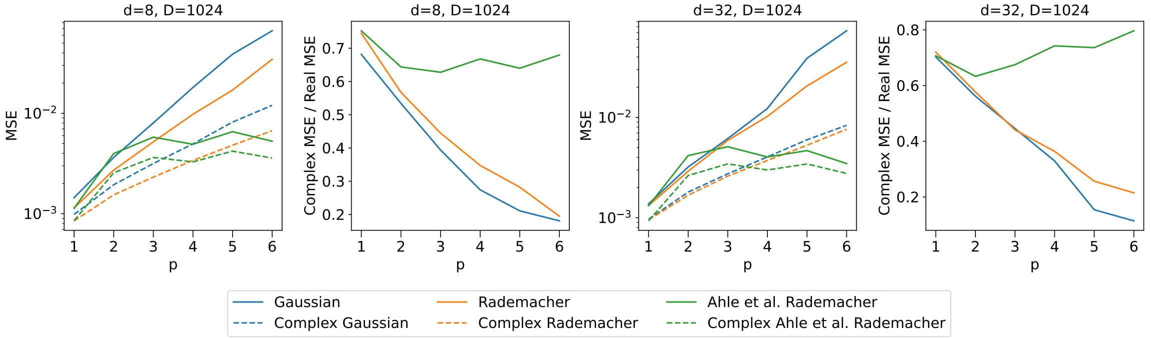

Figure 2: This plot shows the mean squared error $\mathbb{E}[|\hat{k}(\boldsymbol{x},\boldsymbol{y}) - (\boldsymbol{x}^\top \boldsymbol{y})^p|^2]$ for different values of the degree p as well as the mean squared error ratio of the complex sketches over their real analogues. We sample 1,000 independent vector pairs $(\boldsymbol{x},\boldsymbol{y}) \in \mathbb{R}^d \times \mathbb{R}^d$ with $\boldsymbol{x} = |\tilde{\boldsymbol{x}}|/\|\tilde{\boldsymbol{x}}\|, \boldsymbol{y} = |\tilde{\boldsymbol{y}}|/\|\tilde{\boldsymbol{y}}\|$ and $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}_d)$. The mean is then taken over 1,000 independent constructions of the approximate kernel $\hat{k}(\boldsymbol{x},\boldsymbol{y})$, and over every input pair $(\boldsymbol{x},\boldsymbol{y})$.

In Fig. 2 we also included the tree construction in Ahle et al. (2020, Alg. 1) with (complex) Rademacher sketches as nodes. This approach has an improved scaling w.r.t. p compared to our method; however, for commonly used degrees $p = 2, 3$ our method outperforms it. Another interesting observation from the figure is that Ahle et al. (2020) does not benefit from complex sketches in the same way as our approach, even though this yields an error ratio below one. Based on these results, we recommend our approach over the tree method by Ahle et al. (2020) for low degrees $p = 2, 3$ as it is competitive and easier to implement. In § 6 we provide more evidence to support these results.

4. Structured Polynomial Sketches

We study here *structured* polynomial sketches and their extensions with complex features. In Section 3, we studied polynomial sketches in Eq. (5) (or Eq. (10) for complex extensions), where the $p \times D$ random vectors $\boldsymbol{w}_{i,\ell} \in \mathbb{R}^d$ ($i = 1, \ldots, p$, $\ell = 1, \ldots, D$) are generated in an i.i.d. manner. By putting a structural constraint on these vectors, one can construct more efficient random features with a lower variance. Moreover, such a structural constraint leads to a computational advantage, as the imposed structure may be used for implementing an efficient algorithm for fast matrix multiplication.

We consider structured polynomial sketches known as *TensorSRHT (Tensor Subsampled Randomized Hadamard Transform)*. Tropp (2011) studied TensorSRHT for $p = 1$ (linear case) and Hamid et al. (2014); Ahle et al. (2020) extended it[10] to arbitrary polynomial degrees p. Ahle et al. (2020) introduced the name TensorSRHT, which refers to the fact that it implicitly sketches from the d^p-dimensional space of tensorized inputs $\boldsymbol{x}^{(p)}$, as shown in Eq. (7).

[10] The sketches proposed by Hamid et al. (2014) and Ahle et al. (2020) are similar but not exactly the same. Hamid et al. (2014) uses $p \times B$ independent linear SRHT sketches (see Tropp, 2011), where $B := \lceil \frac{D}{d} \rceil$ is the number of SRHT blocks per degree. The elements of these sketches are then shuffled across degrees and blocks, and the blocks are multiplied elementwise over p. Ahle et al. (2020) compute only p independent sketches and subsample from their tensor product instead.

In Section 4.1, we introduce TensorSRHT with real features, and present its extension using complex features in Section 4.2. We then make a comparison between the real and complex versions of TensorSRHT in Section 4.3.

Note that the TensorSRHT algorithm presented here is a slight modification[11] of the one proposed by Hamid et al. (2014). We show that our modification still yields unbiased approximations of polynomial kernels. It further allows us to derive the variance of the sketch in closed form, which has not been done in any of the previous works. We show, for the first time, that TensorSRHT is more efficient than Rademacher sketches for odd p.

4.1 Real TensorSRHT

TensorSRHT imposes an orthogonality constraint on the vectors $\boldsymbol{w}_{i,1}, \dots, \boldsymbol{w}_{i,D}$ through predefined structured matrices, specifically *unnormalized Hadamard matrices.* Let $n := 2^m$ with $m \in \mathbb{N}$, and define $\boldsymbol{H}_n \in \{1, -1\}^{n \times n}$ to be the unnormalized Hadamard matrix, which is recursively defined as

$$\boldsymbol{H}_{2n} := \begin{bmatrix} \boldsymbol{H}_n & \boldsymbol{H}_n \\ \boldsymbol{H}_n & -\boldsymbol{H}_n \end{bmatrix}, \quad \text{with} \quad \boldsymbol{H}_2 := \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{24}$$

From now on, we always use $\boldsymbol{H}_d \in \{1, -1\}^{d \times d}$ with d being the dimensionality of input vectors, assuming $d = 2^m$ for some $m \in \mathbb{N}$. If $d \neq 2^m$ for any $m \in \mathbb{N}$, we pad 0 to input vectors until their dimensionality becomes 2^m for some $m \in \mathbb{N}$. For $i = 1, \dots, d$, let $\boldsymbol{h}_i \in \{1, -1\}^d$ be the i-th column of $\boldsymbol{H}_d$, i.e.,

$$\boldsymbol{H}_d = (\boldsymbol{h}_1, \dots, \boldsymbol{h}_d) \in \{1, -1\}^{d \times d}.$$

Note that we have $\boldsymbol{H}_d \boldsymbol{H}_d^\top = \boldsymbol{H}_d^\top \boldsymbol{H}_d = d\boldsymbol{I}_d$, which implies that distinct columns (and rows) of $\boldsymbol{H}_d$ are orthogonal to each other, i.e., $\boldsymbol{h}_i^\top \boldsymbol{h}_j = 0$ for $i \neq j$.

Case $D = d$. For ease of explanation, suppose here that the number D of random features is equal to the dimensionality d of input vectors: $D = d$. For $i = 1, \dots, p$, define $\boldsymbol{w}_i \in \mathbb{R}^d$ as a random vector whose elements are i.i.d. Rademacher random variables:

$$\boldsymbol{w}_i := (w_{i,1}, \dots, w_{i,d})^\top \in \mathbb{R}^d, \quad w_{i,j} \overset{i.i.d.}{\sim} \text{unif}(\{1, -1\}) \quad (j = 1, \dots, d)$$

Consider a random permutation of the indices $\pi : \{1, \dots, d\} \to \{1, \dots, d\}$, and let

$$\pi(1), \dots, \pi(d)$$

be the permuted indices. For $i = 1, \dots, p$ and $\ell = 1, \dots, D$, we then define a random vector $\boldsymbol{s}_{i,\ell} \in \mathbb{R}^d$ as the Hadamard product (i.e., element-wise product) of the Rademacher vector $\boldsymbol{w}_i$ and the permuted column $\boldsymbol{h}_{\pi(\ell)}$ of the Hadamard matrix:

$$\boldsymbol{s}_{i,\ell} := \boldsymbol{w}_i \circ \boldsymbol{h}_{\pi(\ell)} = (w_{i,1} h_{\pi(\ell),1}, \dots, w_{i,d} h_{\pi(\ell),d})^\top \in \mathbb{R}^d, \tag{25}$$

where $h_{\pi(\ell),j}$ denotes the j-th element of $\boldsymbol{h}_{\pi(\ell)}$.

[11] Instead of permuting elements across degrees and blocks, we only permute the elements inside each block as it is done for SRHT (see Tropp, 2011). Our sketch and the one proposed by Ahle et al. (2020) are equivalent when $D \leq d$ and different when $D > d$.

Because of the orthogonality of the columns $\boldsymbol{h}_1, \dots, \boldsymbol{h}_d$ of the Hadamard matrix $\boldsymbol{H}_d$, the random weight vectors $\boldsymbol{s}_{i,1}, \dots, \boldsymbol{s}_{i,d}$ are orthogonal to each other almost surely: for $\ell \neq m$, we have

$$\boldsymbol{s}_{i,\ell}^\top \boldsymbol{s}_{i,m} = (\boldsymbol{w}_i \circ \boldsymbol{h}_{\pi(\ell)})^\top (\boldsymbol{w}_i \circ \boldsymbol{h}_{\pi(m)}) = \sum_{j=1}^{d} w_{i,j}^2 \boldsymbol{h}_{\pi(\ell),j} \boldsymbol{h}_{\pi(m),j} = \boldsymbol{h}_{\pi(\ell)}^\top \boldsymbol{h}_{\pi(m)} = 0.$$

Note also that, given the permutation $\pi(1), \dots, \pi(d)$, the elements of each random vector $\boldsymbol{s}_{i,\ell}$ in (25) are i.i.d. Rademacher variables.

Finally, we define a random feature map $\Phi_{\mathcal{R}}(\boldsymbol{x}) : \mathbb{R}^d \to \mathbb{R}^D$ for the case $D = d$ as

$$\Phi_{\mathcal{R}}(\boldsymbol{x}) := \frac{1}{\sqrt{D}} \left[(\prod_{i=1}^{p} \boldsymbol{s}_{i,1}^\top \boldsymbol{x}), \dots, (\prod_{i=1}^{p} \boldsymbol{s}_{i,d}^\top \boldsymbol{x}) \right]^\top \in \mathbb{R}^d, \tag{26}$$

which defines an approximate kernel as

$$\hat{k}_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y}) := \Phi_{\mathcal{R}}(\boldsymbol{x})^\top \Phi_{\mathcal{R}}(\boldsymbol{y}) = \frac{1}{D} \sum_{\ell=1}^{D} \Phi(\boldsymbol{x})_{\mathcal{R},\ell} \Phi(\boldsymbol{y})_{\mathcal{R},\ell}$$

where $\Phi(\cdot)_{\mathcal{R},\ell}$ denotes the ℓ-th element of $\Phi_{\mathcal{R}}(\cdot)$.

The orthogonality of the weight vectors in Eq. (25) leads to *negative covariances* between the terms $\Phi(\boldsymbol{x})_{\mathcal{R},\ell}\Phi(\boldsymbol{y})_{\mathcal{R},\ell}$ and $\Phi(\boldsymbol{x})_{\mathcal{R},m}\Phi(\boldsymbol{y})_{\mathcal{R},m}$ with distinct indices $\ell \neq m$ in the approximate kernel. These negative covariances decrease the overall variance of the approximate kernel, as we will show later in Theorem 5 and Appendix C.1

Case $D \neq d$. We now explain the case $D \neq d$. If $D < d$, we first compute the feature map in Eq. (26) and keep the first D components of it. If $D > d$, we independently generate the feature map in Eq. (26) $B := \lceil \frac{D}{d} \rceil$ times and concatenate the resulting B vectors to obtain a Bd-dimensional feature map, and then discard the redundant last $Bd - D$ components of it to obtain a D-dimensional feature map. In this way, we can obtain a D-dimensional feature map for arbitrary $D \in \mathbb{N}$, which we write as

$$\Phi_{\mathcal{R}}(\boldsymbol{x}) := \frac{1}{\sqrt{D}} \left[(\prod_{i=1}^{p} \boldsymbol{s}_{i,1}^\top \boldsymbol{x}), \dots, (\prod_{i=1}^{p} \boldsymbol{s}_{i,D}^\top \boldsymbol{x}) \right]^\top \in \mathbb{R}^D. \tag{27}$$

The entire procedure for constructing the structured polynomial sketch in Eq. (27) is outlined in Algorithm 1, where we also cover the complex-valued case discussed later.

In Algorithm 1, we use an equivalent matrix formulation, since it enables the Fast Walsh-Hadamard transform by employing the associativity, and thus the feature map can be computed much faster. To explain this more precisely, let $\boldsymbol{D}_i := \mathrm{diag}(w_{i1}, \dots, w_{id}) \in \mathbb{R}^{d \times d}$ be a diagonal matrix whose diagonal entries $w_{i1}, \dots, w_{id} \in \{1, -1\}$ are i.i.d. Rademacher random variables, and $\boldsymbol{P}_\pi := (\boldsymbol{e}_{\pi(1)}, \dots, \boldsymbol{e}_{\pi(d)}) \in \mathbb{R}^{d \times d}$ be a permutation matrix, where $\boldsymbol{e}_{\pi(\ell)} \in \mathbb{R}^d$ is a vector whose $\pi(\ell)$-th element is 1 and other elements are 0. We can then compute

$$(\boldsymbol{s}_{i,1}^\top \boldsymbol{x}, \dots, \boldsymbol{s}_{i,d}^\top \boldsymbol{x}) = \boldsymbol{x}^\top (\boldsymbol{s}_{i,1}, \dots, \boldsymbol{s}_{i,d}) = \boldsymbol{x}^\top (\boldsymbol{D}_i \boldsymbol{H}_d \boldsymbol{P}_\pi) = \Big((\boldsymbol{x}^\top \boldsymbol{D}_i) \boldsymbol{H}_d \Big) \boldsymbol{P}_\pi$$

```
Algorithm 1: Real and Complex TensorSRHT
  Result: A feature map $\Phi_{\mathcal{R}/\mathcal{C}}(\boldsymbol{x})$
  Pad $\boldsymbol{x}$ with zeros so that $d$ becomes a power of 2 ;
  Let $B = \lceil \frac{D}{d} \rceil$ be the number of stacked projection blocks ;
  forall $b \in \{1, \dots, B\}$ do
      forall $i \in \{1, \dots, p\}$ do
          Real case Generate a random vector $\mathbf{w}_i = (w_{i,1}, \dots, w_{i,d})^\top \in \mathbb{R}^d$ as
            $w_{i,1}, \dots, w_{i,d} \overset{i.i.d}{\sim} \mathrm{unif}(\{1, -1\})$, and define a diagonal matrix
            $\boldsymbol{D}_i := \mathrm{diag}(\boldsymbol{w}_i) \in \mathbb{R}^{d \times d}$;
          Complex case Generate a random vector $\mathbf{z}_i = (z_{i,1}, \dots, z_{i,d})^\top \in \mathbb{C}^d$ as
            $z_{i,1}, \dots, z_{i,d} \overset{i.i.d}{\sim} \mathrm{unif}(\{1, -1, \mathrm{i}, -\mathrm{i}\})$, and define a diagonal matrix
            $\boldsymbol{D}_i := \mathrm{diag}(\boldsymbol{w}_i) \in \mathbb{C}^{d \times d}$ ;
          Randomly permute the indices $1, \dots, d$ to $\pi(1), \dots, \pi(d)$ ;
          Let $\boldsymbol{P}_\pi := (\boldsymbol{e}_{\pi(1)}, \dots, \boldsymbol{e}_{\pi(d)}) \in \mathbb{R}^{d \times d}$, where $\boldsymbol{e}_{\pi(\ell)} \in \mathbb{R}^d$ is a vector whose
            $\pi(\ell)$-th element is 1 and other elements are 0 $(\ell = 1, \dots, d)$ ;
          Let $(\boldsymbol{s}_{i,1}, \dots, \boldsymbol{s}_{i,d}) := \boldsymbol{D}_i \boldsymbol{H}_d \boldsymbol{P}_\pi$ ;
      end
      Compute $\Phi_b(\boldsymbol{x}) := \sqrt{1/D}[(\prod_{i=1}^p \boldsymbol{s}_{i,1}^\top \boldsymbol{x}), \dots, (\prod_{i=1}^p \boldsymbol{s}_{i,d}^\top \boldsymbol{x})]^\top$ ;
  end
  Concatenate the elements of $\Phi_1(\boldsymbol{x}), \dots, \Phi_B(\boldsymbol{x})$ to yield a single projection vector
    $\Phi_{\mathcal{R}/\mathcal{C}}(\boldsymbol{x})$ and keep the first $D$ entries ;
```

by 1) first computing $\boldsymbol{x}^\top \boldsymbol{D}_i$, 2) then multiplying the Hadamard matrix $\boldsymbol{H}_d$ using the Fast Walsh-Hadamard transform, and 3) lastly multiplying the permutation matrix $\boldsymbol{P}_\pi$, which is more efficient than first precomputing $\boldsymbol{D}_i \boldsymbol{H}_d \boldsymbol{P}_\pi$ and then multiplying $\boldsymbol{x}^\top$. In this way, thanks to the Fast Walsh-Hadamard transform, $(\boldsymbol{s}_{i,1}^\top \boldsymbol{x}, \dots, \boldsymbol{s}_{i,d}^\top \boldsymbol{x})$ can be computed in $\mathcal{O}(d \log d)$ instead of $\mathcal{O}(d^2)$ (Fino and Algazi, 1976). The total computational complexity is therefore $\mathcal{O}(pD \log d)$ and the memory requirement is $\mathcal{O}(pD)$, and this is a computational advantage of TensorSRHT.

The feature map in Eq. (27) induces an approximate kernel $\hat{k}_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y}) = \Phi_{\mathcal{R}}(\boldsymbol{x})^\top \Phi_{\mathcal{R}}(\boldsymbol{y})$. The following proposition summarizes that this approximate kernel is unbiased with respect to the target polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y})^p$. As mentioned earlier, TensorSRHT discussed here is slightly different from the existing versions. Therefore this result is novel in its own right. The result follows from Proposition 9 in the next subsection, so we omit the proof.

Proposition 4 *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ be arbitrary, and $\hat{k}_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y}) = \Phi_{\mathcal{R}}(\boldsymbol{x})^\top \Phi_{\mathcal{R}}(\boldsymbol{y})$ be the approximate kernel with $\Phi_{\mathcal{R}}(\boldsymbol{x}), \Phi_{\mathcal{R}}(\boldsymbol{y}) \in \mathbb{R}^D$ given by the random feature map in Eq.* (27). *Then we have $\mathbb{E}[\hat{k}_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y})] = (\boldsymbol{x}^\top \boldsymbol{y})^p$.*

We next study the variance of the approximate kernel given by TensorSRHT, which is the mean squared error of the approximate kernel since it is unbiased as shown above. The following theorem provides a closed form expression for the variance, whose proof is given for the more general complex case in Appendix B.2. It is a novel result and extends Choromanski et al. (2017, Theorem 3.3) to the setting $p > 1$ and $D > d$.

Theorem 5 (Variance of Real TensorSRHT) *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ be arbitrary, and $\hat{k}_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y}) = \Phi_{\mathcal{R}}(\boldsymbol{x})^\top \Phi_{\mathcal{R}}(\boldsymbol{y})$ be the approximate kernel with $\Phi(\boldsymbol{x}), \Phi(\boldsymbol{y}) \in \mathbb{R}^D$ given by the random feature map in Eq. (27). Then we have*

$$\mathbb{V}\left[\hat{k}_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y})\right] = \underbrace{\frac{V_{\text{Rad}}^{(p)}}{D}}_{(A)} - \underbrace{\frac{c(D,d)}{D^2}\left[(\boldsymbol{x}^\top \boldsymbol{y})^{2p} - \left((\boldsymbol{x}^\top \boldsymbol{y})^2 - \frac{V_{\text{Rad}}^{(1)}}{d-1}\right)^p\right]}_{(B)}, \tag{28}$$

where $V_{\text{Rad}}^{(p)} \geq 0$ and $V_{\text{Rad}}^{(1)} \geq 0$ are the variances of the real Rademacher sketch with a single feature in Eq. (18) with generic $p \in \mathbb{N}$ and $p = 1$, respectively, and $c(D,d) \in \mathbb{N}$ is defined by

$$c(D,d) := \lfloor D/d \rfloor d(d-1) + \operatorname{mod}(D,d)(\operatorname{mod}(D,d) - 1). \tag{29}$$

Remark 6 *The constant $c(D,d)$ in Eq. (29) is the number of pairs of indices $\ell, \ell' = 1, \ldots, D$ with $\ell \neq \ell'$ for which the covariance of the weight vectors $\boldsymbol{s}_{i,\ell}$ and $\boldsymbol{s}_{i,\ell'}$ in Eq. (27) is non-zero (see the proof in Appendix B.2 for details). If $D = Bd$ for some $B \in \mathbb{N}$, this constant simplifies to $c(D,d) = Bd(d-1)$, and the variance in Eq. (28) becomes*

$$\mathbb{V}\left[\hat{k}_{\mathcal{R}}(\boldsymbol{x}, \boldsymbol{y})\right] = \frac{1}{D} V_{\text{Rad}}^{(p)} - \frac{d-1}{D}\left[(\boldsymbol{x}^\top \boldsymbol{y})^{2p} - \left((\boldsymbol{x}^\top \boldsymbol{y})^2 - \frac{V_{\text{Rad}}^{(1)}}{d-1}\right)^p\right].$$

An interesting subcase is $p = 1$, for which the variance becomes zero. Thus, setting $D \in \{kd \mid k \in \mathbb{N}\}$ for $p = 1$ is equivalent to using the linear kernel with the original inputs.

Theorem 5 enables understanding the condition under which TensorSRHT has a smaller variance than the unstructured Rademacher sketch in Eq. (5). Note that the term (A) in Eq. (28) is the variance of the approximate kernel with the Rademacher sketch with D features. On the other hand, the term (B) in Eq. (28) can be interpreted as the effect of the structured sketch. The term (B) always becomes non-negative when p is odd, and thus the overall variance of TensorSRHT becomes smaller than the Rademacher sketch, as summarized in the following corollary. Thus, when p is odd, TensorSRHT should be preferred over the Rademacher sketch.

Corollary 7 *Let $p \in \mathbb{N}$ be odd. Then, for all input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, the variance of the approximate kernel with TensorSRHT in Eq. (28) is smaller or equal to the variance of the approximate kernel with the Rademacher sketch:*

$$\frac{V_{\text{Rad}}^{(p)}}{D} - \frac{c(D,d)}{D^2}\left[(\boldsymbol{x}^\top \boldsymbol{y})^{2p} - \left((\boldsymbol{x}^\top \boldsymbol{y})^2 - \frac{V_{\text{Rad}}^{(1)}}{d-1}\right)^p\right] \leq \frac{V_{\text{Rad}}^{(p)}}{D}$$

Proof Since, $V_{\text{Rad}}^{(1)} \geq 0$, we have $(\boldsymbol{x}^\top \boldsymbol{y})^2 - \frac{1}{d-1} V_{\text{Rad}}^{(1)} \leq (\boldsymbol{x}^\top \boldsymbol{y})^2$. For odd p this leads to $\left((\boldsymbol{x}^\top \boldsymbol{y})^2 - \frac{1}{d-1} V_{\text{Rad}}^{(1)}\right)^p \leq (\boldsymbol{x}^\top \boldsymbol{y})^{2p}$. The assertion immediately follows. ■

If p is even, on the other hand, the variance of TensorSRHT can be larger than the Rademacher sketch for certain input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. For instance, if $\boldsymbol{x}$ and $\boldsymbol{y}$ are orthogonal, i.e., $\boldsymbol{x}^\top \boldsymbol{y} = 0$, then the variance of TensorSRHT in Eq. (28) is

$$\text{Eq. (28)} = \frac{V_{\text{Rad}}^{(p)}}{D} + \frac{c(D,d)}{D^2} \left(\frac{V_{\text{Rad}}^{(1)}}{d-1} \right)^p \geq \frac{V_{\text{Rad}}^{(p)}}{D}.$$

Therefore, for even p, we do not have a theoretical guarantee for the advantage of TensorSRHT over the Rademacher sketch in terms of their variances. In practice, however, TensorSRHT has often a smaller variance than the Rademacher sketch also for even p, as demonstrated in our experiments described later. Moreover, TensorSRHT has a computational advantage over the Rademacher sketch, thanks to the fast Walsh-Hadamard transform.

Remark 8 *One can straightforwardly derive a probabilistic error bound for TensorSRHT by using Theorem 5 and Chebyshev's inequality. However, deriving an exponential tail bound for TensorSRHT is more involved, because different features in the feature map $\Phi_{\mathcal{R}}(\boldsymbol{x})$ in Eq. (26) are dependent for TensorSRHT and thus applying Bersnstein's inequality is not straightforward. One can find an exponential tail bound for TensorSRHT in Ahle et al. (2020, Lemma 33 in the longer version), while they analyze a slightly different version of TensorSRHT from ours and their bound is a uniform upper bound that holds for all input vectors simultaneously.*

Our variance formula in Eq. (28), which is a novel contribution to the literature, provides a precise characterization of how the variance of the approximate kernel depends on the input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, and shows when TensorSRHT is more advantageous than the Rademacher sketch. Moreover, as the variance formula can be computed in practice, it can be used for designing an objective function for a certain optimization problem, as we do in Section 5.3 for designing a data-driven approach to feature construction.

4.2 Complex-valued TensorSRHT

We present here a generalization of TensorSRHT by allowing for complex features. To this end, let $z \in \mathbb{C}$ be a random variable such that (i) $|z| = 1$ almost surely, (ii) $\mathbb{E}[z] = 0$ and (iii) z is symmetric, i.e., the distributions of z and $-z$ are the same. Define then $\boldsymbol{z}_1, \dots, \boldsymbol{z}_p \in \mathbb{C}^d$ as i.i.d. complex random vectors such that elements of each random vector $\boldsymbol{z}_i$ are i.i.d. realizations of z:

$$\boldsymbol{z}_i = (z_{i,1}, \dots, z_{i,d})^\top \in \mathbb{C}^d, \quad z_{i,j} \overset{i.i.d.}{\sim} P_z \quad (j = 1, \dots, d), \tag{30}$$

where P_z denotes the probability distribution of z.

Let $\pi : \{1, \dots, d\} \to \{1, \dots, d\}$ be a random permutation of indices $1, \dots, d$. For $i = 1, \dots, p$ and $\ell = 1, \dots, D$, we then define a random vector $\boldsymbol{s}_{i,\ell} \in \mathbb{C}^d$ as the Hadamard product of the random vector $\boldsymbol{z}_i$ in (30) and the permuted column $\boldsymbol{h}_{\pi(\ell)}$ of the Hadamard matrix $\boldsymbol{H}_d$:

$$\boldsymbol{s}_{i,\ell} := \boldsymbol{z}_i \circ \boldsymbol{h}_{\pi(\ell)} = (z_{i,1} h_{\pi(\ell),1}, \dots, z_{i,d} h_{\pi(\ell),d})^\top \in \mathbb{C}^d, \tag{31}$$

With these weight vectors $\boldsymbol{s}_{i,\ell}$, we define a random feature map exactly in the same way as the feature map in Eq. (27) for the real TensorSRHT in Section 4.1. We define the resulting feature map $\Phi_{\mathcal{C}} : \mathbb{R}^d \to \mathbb{C}^D$ by

$$\Phi_{\mathcal{C}}(\boldsymbol{x}) := \frac{1}{\sqrt{D}} \left[(\prod_{i=1}^{p} \boldsymbol{s}_{i,1}^{\top} \boldsymbol{x}), \dots, (\prod_{i=1}^{p} \boldsymbol{s}_{i,D}^{\top} \boldsymbol{x}) \right]^{\top} \in \mathbb{C}^D. \tag{32}$$

We call this feature construction *complex TensorSRHT.*

Admissible examples of the distribution P_z in Eq. (30) include: (1) the uniform distribution on $\{1, -1\}$; (2) the uniform distribution on $\{1, -1, \mathrm{i}, -\mathrm{i}\}$; and (3) the uniform distribution on the unit circle in $\mathbb{C}^d$. Example (1) is where z is a real Rademacher random variable, and in this case, the complex TensorSRHT coincides with the real TensorSRHT. Thus, the complex TensorSRHT is a strict generalization of the real TensorSRHT.

We first show that the complex TensorSRHT provides an unbiased approximation of the polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^{\top} \boldsymbol{y})^p$. Since the real TensorSRHT is a special case, its unbiasedness follows from this result.

Proposition 9 *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ be arbitrary, and $\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y}) = \Phi_{\mathcal{C}}(\boldsymbol{x})^{\top} \overline{\Phi_{\mathcal{C}}(\boldsymbol{y})}$ be the approximate kernel with $\Phi_{\mathcal{C}}(\boldsymbol{x}), \Phi_{\mathcal{C}}(\boldsymbol{y}) \in \mathbb{C}^D$ given by the random feature map in Eq.* (32)*. Then we have $\mathbb{E}[\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y})] = (\boldsymbol{x}^{\top} \boldsymbol{y})^p$.*

Proof We first show $\mathbb{E}[\boldsymbol{s}_{i,\ell} \overline{\boldsymbol{s}_{i,\ell}}^{\top}] = \boldsymbol{I}_d$ for all $i = 1, \dots, p$ and $\ell = 1, \dots, D$. This follows from the fact that, for all $t, u = 1, \dots, d$, we have

$$\mathbb{E}[(\boldsymbol{s}_{i,\ell} \overline{\boldsymbol{s}_{i,\ell}}^{\top})_{tu}] = \mathbb{E}[z_{i,t} h_{\pi(\ell),t} \overline{z_{i,u}} h_{\pi(\ell),u}] = \begin{cases} \mathbb{E}[|z_{i,t}|^2] \mathbb{E}[h_{\pi(\ell),t}^2] = 1 \quad (\text{if} \ \ t = u), \\ \mathbb{E}[z_{i,t}] \mathbb{E}[\overline{z_{i,u}}] \mathbb{E}[h_{\pi(\ell),t} h_{\pi(\ell),u}] = 0 \quad (\text{if} \ \ t \neq u), \end{cases} .$$

Using this, we have

$$\mathbb{E}\left[\Phi_{\mathcal{C}}(\boldsymbol{x})^{\top} \overline{\Phi_{\mathcal{C}}(\boldsymbol{y})} \right] = \mathbb{E}\left[\frac{1}{D} \sum_{\ell=1}^{D} \prod_{i=1}^{p} \boldsymbol{x}^{\top} \boldsymbol{s}_{i,\ell} \overline{\boldsymbol{s}_{i,\ell}}^{\top} \boldsymbol{y} \right] = \frac{1}{D} \sum_{\ell=1}^{D} \prod_{i=1}^{p} \boldsymbol{x}^{\top} \mathbb{E}[\boldsymbol{s}_{i,\ell} \overline{\boldsymbol{s}_{i,\ell}}^{\top}] \boldsymbol{y} = (\boldsymbol{x}^{\top} \boldsymbol{y})^p$$

■

We now study the variance of the approximate kernel given by the complex TensorSRHT in Eq. (32). To this end, we use the same notation as Theorem 2 to write the real and imaginary parts of the random variable z as $z = a + \mathrm{i}b$ with real-valued random variables $a, b \in \mathbb{R}$. The proof of the following theorem is provided in Appendix B.2.

Theorem 10 (Variance of Complex TensorSRHT) *Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ be arbitrary, and $\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y}) = \Phi_{\mathcal{C}}(\boldsymbol{x})^{\top} \overline{\Phi_{\mathcal{C}}(\boldsymbol{y})}$ be the approximate kernel with $\Phi_{\mathcal{C}}(\boldsymbol{x}), \Phi_{\mathcal{C}}(\boldsymbol{y}) \in \mathbb{C}^D$ given by the complex random feature map in Eq.* (32)*. For the random variable z defining Eq.* (30)*, write $z = a + \mathrm{i}b$ with $a, b \in \mathbb{R}$, and suppose that*

$$\mathbb{E}[ab] = 0, \quad \mathbb{E}[a^2] = q, \quad \mathbb{E}[b^2] = 1 - q \quad \textit{where} \quad 0 \leq q \leq 1.$$

Then we have

$$\mathbb{V}[\hat{k}_{\mathcal{C}}(\boldsymbol{x},\boldsymbol{y})] = \underbrace{\frac{V_q^{(p)}}{D}}_{(A)} - \underbrace{\frac{c(D,d)}{D^2}\left[(\boldsymbol{x}^\top\boldsymbol{y})^{2p} - \left((\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{V_q^{(1)}}{d-1}\right)^p\right]}_{(B)}, \tag{33}$$

where $V_q^{(p)} \geq 0$ and $V_q^{(1)} \geq 0$ are Eq. (17) *with the considered value of p and $p = 1$, respectively, and $c(D,d) \in \mathbb{N}$ is defined in* (29)*.*

Regarding Theorem 10, we make the following observations.

- The case $q = 1$ recovers Theorem 5 on the real TensorSRHT, where $z \in \{1, -1\}$ is a Rademacher random variable. The case $q = 1/2$ is the complex TensorSRHT with, for instance, P_z being the uniform distribution on $\{1, -1, \mathrm{i}, -\mathrm{i}\}$ or on the unit circle in $\mathbb{C}$. Other values of $q \in [0,1]$ can also be considered, but we do not discuss them further.

- The first term (A) in Eq. (33) is the variance of the unstructured polynomial sketch in Eq. (10) with D features, since $V_q^{(p)}$ is its variance with a single feature ($D = 1$) in Eq. (17). The second term (B) in Eq. (33) is the effect of using the structured sketch. The quantity $V_q^{(1)}$ is the variance of the unstructured sketch in Eq. (10) with a single feature in Eq. (17) with $p = 1$.

- As for the real case, the variance (33) becomes zero when $p = 1$ and $D \in \{kd \mid k \in \mathbb{N}\}$.

As we discussed for the real TensorSRHT in Corollary 7, Theorem 10 enables understanding a condition under which the complex TensorSRHT is advantageous over the corresponding unstructured complex sketch in Eq. (10). As for the real case, the condition is that the degree p of the polynomial kernel is *odd*, as stated in the following.

Corollary 11 *Let $p \in \mathbb{N}$ be odd. Then, for all input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, the variance of the approximate kernel with the complex TensorSRHT in Eq.* (33) *is smaller or equal to the variance of the approximate kernel with the corresponding unstructured polynomial sketch:*

$$\frac{V_q^{(p)}}{D} - \frac{c(D,d)}{D^2}\left[(\boldsymbol{x}^\top\boldsymbol{y})^{2p} - \left((\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{V_q^{(1)}}{d-1}\right)^p\right] \leq \frac{V_q^{(p)}}{D}$$

Proof Since, $V_q^{(1)} \geq 0$, we have $(\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{1}{d-1}V_q^{(1)} \leq (\boldsymbol{x}^\top\boldsymbol{y})^2$. For odd p this leads to $\left((\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{1}{d-1}V_q^{(1)}\right)^p \leq (\boldsymbol{x}^\top\boldsymbol{y})^{2p}$. The assertion immediately follows. ■

As discussed for the real case, if p is even, the variance of the complex TensorSRHT can be larger than the corresponding unstructured sketch for certain input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ (e.g., when $\boldsymbol{x}^\top\boldsymbol{y} = 0$). Empirically, however, the complex TensorSRHT often has a smaller variance also for even p, as we demonstrate later.

4.3 Comparing the Real and Complex TensorSRHT

Let us now compare the real and complex TensorSRHT. To make the discussion clearer, suppose that the number of random features satisfies $D = Bd$ for some $B \in \mathbb{N}$, as in Remark 6. Then the variance formula in Eq. (33) simplifies to

$$\mathbb{V}[\hat{k}_{\mathcal{C}}(\boldsymbol{x}, \boldsymbol{y})] = \underbrace{\frac{V_q^{(p)}}{D}}_{(A)} - \underbrace{\frac{d-1}{D}\left[(\boldsymbol{x}^\top \boldsymbol{y})^{2p} - \left((\boldsymbol{x}^\top \boldsymbol{y})^2 - \frac{V_q^{(1)}}{d-1}\right)^p\right]}_{(B)}. \tag{34}$$

Recall that setting $q = 1$ and $q = 1/2$ recover the variances of real and complex TensorSRHT, respectively. Thus, let us compare these two cases. We make the following observations:

- As discussed in Section 3.3, it holds that $V_{1/2}^{(p)} \leq V_1^{(p)}$ and $V_{1/2}^{(1)} \leq V_1^{(1)}$ given that the input vectors $\boldsymbol{x} = (x_1, \ldots, x_d), \boldsymbol{y} = (y_1, \ldots, y_d)$ satisfy the inequality in Eq. (22), i.e., $\sum_{i \neq j} x_i x_j y_i y_j \geq 0$, which is satisfied when $\boldsymbol{x}$ and $\boldsymbol{y}$ are non-negative vectors.

- Thus, if Eq. (22) is satisfied, the first term (A) becomes smaller for $q = 1/2$ (complex case) than $q = 1$ (real case). On the other hand, if p is odd, the second term (B) becomes smaller for $q = 1/2$ than $q = 1$; thus, the variance reduction (i.e., $-(B)$) is smaller for $q = 1/2$ than $q = 1$.

The above observations suggest that, even when Eq. (22) is satisfied, whether the complex TensorSRHT ($q = 1/2$) has a smaller variance than the real TensorSRHT ($q = 1$) depends on the balance between the two terms (A) and (B) and on the properties of the input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$. We have not been able to provide a theoretical characterization of exact situations where the complex TensorSRHT has a smaller variance than the real TensorSRHT.

To complement the lack of a theoretical characterization, we performed experiments to compare the variances of real and complex TensorSRHT, whose results are shown in Fig. 3. We evaluated the variance formula in Eq. (34) for $q = 1$ (real) and $q = 1/2$ (complex), for 1000 pairs of input vectors $\boldsymbol{x}, \boldsymbol{y}$ randomly sampled from a given dataset (EEG, CIFAR 10 ResNet34 features, MNIST and Gisette). For each pair $\boldsymbol{x}, \boldsymbol{y}$, we computed the ratio of Eq. (34) with $q = 1/2$ divided by Eq. (34) with $q = 1$, and Fig. 3 shows the empirical cumulative distribution function of this ratio for the 4 datasets. In these datasets, the input vectors are nonnegative.

Fig. 3 shows that, for 100%, 100%, 97.8%, and 100% of the cases of the 4 datasets, respectively, the variance of the complex TensorSRHT is smaller than that of the real TensorSRHT. Moreover, the ratio of the variances tends to be even smaller for a larger value of p. These results suggest that the complex TensorSRHT is effective in reducing the variance of the real TensorSRHT, and the variance reduction is more significant for a larger value p of the polynomial degree. We leave a theoretical analysis for explaining this improvement of the complex TensorSRHT for future work.

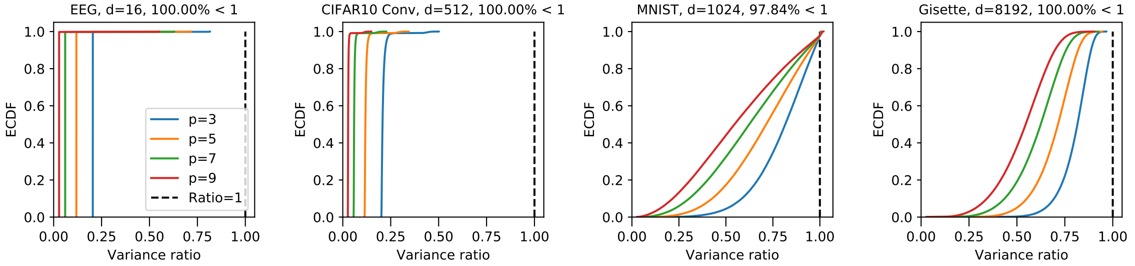

Figure 3: Empirical cumulative distribution of pairwise ratios Var(Compl. TensorSRHT) / Var(Real TensorSRHT) on a subsample (1000 samples) of four different datasets (EEG, CIFAR10 ResNet34 features, MNIST, Gisette) with unit-normalized data where $D = d$. The datasets are not zero-centered and therefore entirely positive.

5. Approximating Dot Product Kernels

We discuss here how polynomial sketches described so far can be used for approximating more general *dot product kernels*, i.e., kernels whose values depend only on the inner product of input vectors.

In Sections 5.1 and 5.2, we first review a key result on the Maclaurin expansion of dot product kernels and the resulting random sketch approach by Kar and Karnick (2012), and show how the polynomial sketches described so far can be used. In Section 5.3, we then introduce a data-driven optimization approach to improving the random sketches based on the Maclaurin expansion. In Section 5.4, we describe how to apply this approach for approximating the Gaussian kernel. In Section 5.5, we provide a numerical illustration of the optimization objective.

5.1 Maclaurin Expansion of Dot Product Kernels

Let $\mathcal{X} \subset \mathbb{R}^d$ be a subset, and let $k : \mathcal{X} \times \mathcal{X} :\to \mathbb{R}$ be a positive definite kernel on $\mathcal{X}$. The kernel k is called *dot product kernel*, if there exists a function $f : \mathbb{R} \to \mathbb{R}$ such that

$$k(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{x}^\top \boldsymbol{y}) \quad \text{for all } \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}. \tag{35}$$

Examples of dot product kernels include polynomial kernels $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y} + \nu)^p$ with $\nu \geq 0$ and $p \in \mathbb{N}$, which have been our focus in this paper, and exponential kernels $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(\boldsymbol{x}^\top \boldsymbol{y}/l^2)$ with $l > 0$. Other examples of dot product kernels can be found in, e.g., Smola et al. (2000).

We focus on dot product kernels for which the function f in Eq. (35) is an analytic function whose Maclaurin expansion has non-negative coefficients: $f(x) = \sum_{n=0}^{\infty} a_n x^n$ and $a_n \geq 0$ for $n \in \{0\} \cup \mathbb{N}$. In other words, we consider dot product kernels that can be expanded as

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{n=0}^{\infty} a_n (\boldsymbol{x}^\top \boldsymbol{y})^n \quad \text{for all } \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}, \tag{36}$$

with $a_n \geq 0$ for all $n \in \{0\} \cup \mathbb{N}$.

Many dot product kernels can be expanded as Eq. (36). In fact, Kar and Karnick (2012, Theorem 1) show that, if $\mathcal{X}$ is the unit ball of $\mathbb{R}^d$, the function k of the form of Eq. (35) is positive definite on $\mathcal{X}$ if and only if it can be written as Eq. (36).

We show here a few concrete examples. The polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y} + \nu)^p$ with $p \in \mathbb{N}$ and $\nu \geq 0$ can be expanded as

$$(\boldsymbol{x}^\top \boldsymbol{y} + \nu)^p = \sum_{n=0}^{p} \binom{p}{n} \nu^{p-n} (\boldsymbol{x}^\top \boldsymbol{y})^n, \tag{37}$$

and thus $a_n = \binom{p}{n}\nu^{p-n} \geq 0$ for $n \in \{0, \ldots, p\}$ and $a_n = 0$ for $n > p$ in Eq. (36). The exponential kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(\boldsymbol{x}^\top \boldsymbol{y}/l^2)$ can be expanded as

$$\exp\left(\frac{\boldsymbol{x}^\top \boldsymbol{y}}{l^2}\right) = \sum_{n=0}^{\infty} \frac{1}{n! l^{2n}} (\boldsymbol{x}^\top \boldsymbol{y})^n \tag{38}$$

and thus $a_n = 1/(n! l^{2n})$ for $n \in \mathbb{N}$ in Eq. (36).

Gaussian kernel as a weighted dot product kernel The Gaussian kernel defined as $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|^2/(2l^2))$ with $l > 0$ can be written as a *weighted* exponential kernel:

$$\begin{aligned}
\exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{2l^2}\right) &= \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2l^2}\right) \exp\left(-\frac{\|\boldsymbol{y}\|^2}{2l^2}\right) \exp\left(\frac{\boldsymbol{x}^\top \boldsymbol{y}}{l^2}\right) \\
&= \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2l^2}\right) \exp\left(-\frac{\|\boldsymbol{y}\|^2}{2l^2}\right) \sum_{n=0}^{\infty} \frac{1}{n! l^{2n}} (\boldsymbol{x}^\top \boldsymbol{y})^n, \qquad (39)
\end{aligned}$$

where the second identity uses the Maclaurin expansion of the exponential kernel in Eq. (38). For approximating the Gaussian kernel, Cotter et al. (2011) proposed a finite dimensional feature map based on a truncation of this expansion.

5.2 Random Sketch based on the Maclaurin Expansion

We describe here the approach of Kar and Karnick (2012) on the unbiased approximation of dot product kernels based on the Maclaurin expansion in Eq. (36). We discuss this approach to provide a basis and motivation for our new approach for approximating dot product kernels.

First, we define a probability measure μ on $\{0\} \cup \mathbb{N}$. Kar and Karnick (2012) propose to define μ as

$$\mu(n) \propto c^{-(n+1)}, \quad n \in \{0\} \cup \mathbb{N}, \tag{40}$$

for a constant $c > 1$ (e.g., $c = 2$). Using this probability measure and the Rademacher sketch, Kar and Karnick (2012) propose a doubly stochastic approximation of the dot product kernel in Eq. (36). This approach first generates an i.i.d. sample of size $D \in \mathbb{N}$ from this probability measure μ

$$n_1, \ldots, n_D \overset{i.i.d.}{\sim} \mu \tag{41}$$

and defines D_n for $n \in \{0\} \cup \mathbb{N}$ as the number of times n appears in $n_1, \ldots, n_D$; thus $\sum_{n=0}^{\infty} D_n = D$.

Then, for each $n \in \{0\} \cup \mathbb{N}$ with $D_n > 0$, construct a random feature map $\Phi_n : \mathcal{X} \to \mathbb{R}^{D_n}$ with D_n features of the form in Eq. (5) that provide an unbiased approximation of the polynomial kernel $k_n(\boldsymbol{x}, \boldsymbol{y}) := (\boldsymbol{x}^\top \boldsymbol{y})^n$ of degree n:

$$\mathbb{E}[\Phi_n(\boldsymbol{x})^\top \Phi_n(\boldsymbol{y})] = (\boldsymbol{x}^\top \boldsymbol{y})^n. \tag{42}$$

The original formulation of Kar and Karnick (2012) uses the Rademacher sketch as Φ_n, but one can use other sketches in Sections 3 and 4, such as the Gaussian sketch and TensorSRHT.

Finally, defining a random variable $n^* \sim \mu$, the dot product kernel in Eq. (36) is rewritten and approximated as

$$\begin{aligned} k(\boldsymbol{x}, \boldsymbol{y}) &= \sum_{n=0}^{\infty} a_n (\boldsymbol{x}^\top \boldsymbol{y})^n = \sum_{n=0}^{\infty} \frac{a_n}{\mu(n)} \mu(n) (\boldsymbol{x}^\top \boldsymbol{y})^n = \mathbb{E}_{n^* \sim \mu} \left[\frac{a_{n^*}}{\mu(n^*)} (\boldsymbol{x}^\top \boldsymbol{y})^{n^*} \right] \\ &\approx \frac{1}{D} \sum_{n \in \{n_1, \ldots, n_D\}} D_n \frac{a_n}{\mu(n)} (\boldsymbol{x}^\top \boldsymbol{y})^n = \frac{1}{D} \sum_{n : D_n > 0} D_n \frac{a_n}{\mu(n)} (\boldsymbol{x}^\top \boldsymbol{y})^n \\ &\approx \frac{1}{D} \sum_{n : D_n > 0} D_n \frac{a_n}{\mu(n)} \Phi_n(\boldsymbol{x})^\top \Phi_n(\boldsymbol{y}), \end{aligned} \tag{43}$$

where the first approximation is the Monte Carlo approximation of the expectation $\mathbb{E}_{n^* \sim \mu}$ using the i.i.d. sample in Eq. (41) and the second approximation is using the random feature map in Eq. (42). The approximation in Eq. (43) is unbiased, since the two approximations are statistically independent and both are unbiased.

The first approximation for Eq. (43) can be interpreted as first selecting polynomial degrees $n \in \{0\} \cup \mathbb{N}$ and assigning the number of features D_n to each selected degree, given a budget constraint $D = \sum_{n : D_n > 0} D_n$. While performing these assignments by random sampling as in Eq. (41) makes the approximation in Eq. (43) unbiased, the resulting variance of Eq. (43) can be large. In the next subsection, we introduce a data-driven optimization approach to this feature assignment problem, to achieve a good balance between the bias and variance.

5.3 Optimization for a Truncated Maclaurin Approximation

We develop here an optimization algorithm for selecting the polynomial degrees n and assigning the number of random features to each selected polynomial degree in the Maclaurin sketch in Eq. (43) . The objective function is an estimate of the expected bias and variance of the resulting approximate kernel, and we define it using the variance formulas derived in Sections 3 and 4.

We consider a biased approximation obtained by truncating the Maclaurin expansion in Eq. (36) up to the p-th degree polynomials, where p is to be determined by optimization. Let $D_{\text{total}} \in \mathbb{N}$ be the total number of random features, which is specified by a user. For each $n = 1, \ldots, p$, let $D_n \in \{0\} \cup \mathbb{N}$ be the number of random features for approximating the n-th term $(\boldsymbol{x}^\top \boldsymbol{y})^n$ of the Maclaurin expansion in Eq. (36), such that $\sum_{n=1}^{p} D_n = D_{\text{total}}$. The numbers D_n are to be determined by optimization. Let $\Phi_n : \mathbb{R}^d \to \mathbb{C}^{D_n}$ be a (possibly complex) random feature map defined in Sections 3 and 4 such that $\mathbb{E}[\Phi_n(\boldsymbol{x})^\top \overline{\Phi_n(\boldsymbol{y})}] = (\boldsymbol{x}^\top \boldsymbol{y})^n$ for all $\boldsymbol{x}, \boldsymbol{y} \subset \mathcal{X} \subset \mathbb{R}^d$. Note that Φ_n can be a real-valued feature map, but we use the notation for the complex case since it subsumes the real case.

We then define an approximation to the dot product kernel in Eq. (36) as

$$\hat{k}(\boldsymbol{x},\boldsymbol{y}) := a_0 + \sum_{n=1}^{p} a_n \Phi_n(\boldsymbol{x})^\top \overline{\Phi_n(\boldsymbol{y})}, \quad \boldsymbol{x},\boldsymbol{y} \in \mathcal{X} \tag{44}$$

This approximation is biased, since it ignores the polynomial terms whose degrees are higher than p in the expansion of Eq. (36). One can reduce this bias by increasing p, but this may lead to a higher variance. Therefore, there is a bias-variance trade-off in the choice of p. We describe below how to choose p and the number of features D_n of each random feature map $\Phi_n(\boldsymbol{x}), \Phi_n(\boldsymbol{y}) \in \mathbb{C}^{D_n}$ for $n = 1, \dots, p$.

5.3.1 Optimization Objective

For a given learning task, we are usually provided data points generated from an unknown probability distribution $P(\boldsymbol{x})$ on the input domain $\mathcal{X} \subset \mathbb{R}^d$. The approximate kernel $\hat{k}(\boldsymbol{x},\boldsymbol{y})$ in Eq. (44) should be an accurate approximation of the target kernel $k(\boldsymbol{x},\boldsymbol{y})$ for input vectors $\boldsymbol{x},\boldsymbol{y}$ drawn from this unknown data distribution $P(\boldsymbol{x})$. Therefore, we consider the following *integrated mean squared error* as our objective function:

$$\int\int \mathbb{E}\left[\left(k(\boldsymbol{x},\boldsymbol{y}) - \hat{k}(\boldsymbol{x},\boldsymbol{y})\right)^2\right] dP(\boldsymbol{x})dP(\boldsymbol{y}) \tag{45}$$

$$= \int\int \underbrace{\mathbb{V}[\hat{k}(\boldsymbol{x},\boldsymbol{y})]}_{\text{variance}} dP(\boldsymbol{x})dP(\boldsymbol{y}) + \int\int \underbrace{\left(k(\boldsymbol{x},\boldsymbol{y}) - \mathbb{E}\left[\hat{k}(\boldsymbol{x},\boldsymbol{y})\right]\right)^2}_{\text{bias}^2} dP(\boldsymbol{x})dP(\boldsymbol{y}) \tag{46}$$

where the expectation $\mathbb{E}[\cdot]$ and variance $\mathbb{V}[\cdot]$ are taken with respect to the random feature maps in the approximate kernel in Eq. (44), and the identity follows from the standard bias-variance decomposition.

We study the variance and bias terms in Eq. (46). Let $\delta[D_n > 0]$ be an indicator such that $\delta[D_n > 0] = 1$ if $D_n > 0$ and $\delta[D_n > 0] = 0$ otherwise. Using this indicator, and since the p random feature maps $\Phi_1, \dots, \Phi_p$ in Eq. (44) are statistically independent, the variance term in Eq. (46) can be written as

$$\mathbb{V}\left[\hat{k}(\boldsymbol{x},\boldsymbol{y})\right] = \sum_{n=1}^{p} \delta[D_n > 0]\ a_n^2 \mathbb{V}\left[\Phi_n(\boldsymbol{x})^\top \overline{\Phi_n(\boldsymbol{y})}\right]. \tag{47}$$

Each individual term $\mathbb{V}[\Phi_n(\boldsymbol{x})^\top \overline{\Phi_n(\boldsymbol{y})}]$ in Eq. (47) is the variance of the approximate kernel $\hat{k}_n(\boldsymbol{x},\boldsymbol{y}) := \Phi_n(\boldsymbol{x})^\top \overline{\Phi_n(\boldsymbol{y})}$ for approximating the polynomial kernel $k_n(\boldsymbol{x},\boldsymbol{y}) := (\boldsymbol{x}^\top \boldsymbol{y})^n$ of degree $n = 1, \dots, p$. Therefore, one can explicitly compute $\mathbb{V}[\Phi_n(\boldsymbol{x})^\top \overline{\Phi_n(\boldsymbol{y})}]$ for any given $\boldsymbol{x},\boldsymbol{y} \in \mathbb{R}^d$ using the variance formulas derived in Sections 3 and 4. For the convenience of the reader, we summarize the variance formulas for specific cases in Table 1. Regarding the bias term in Eq. (46), the expectation of the approximate kernel (44) is given by

$$\mathbb{E}\left[\hat{k}(\boldsymbol{x},\boldsymbol{y})\right] = \sum_{n=0}^{p} \delta[D_n > 0]\ a_n(\boldsymbol{x}^\top \boldsymbol{y})^n, \tag{48}$$

since $\mathbb{E}\left[\Phi_n(\boldsymbol{x})^\top \overline{\Phi_n(\boldsymbol{y})}\right] = (\boldsymbol{x}^\top \boldsymbol{y})^n$ for $n = 1, \dots, p$ with $D_n > 0$.

Sketch	Variance
Real Gaussian	$D^{-1}\Big[\Big(\\|\boldsymbol{x}\\|^2\\|\boldsymbol{y}\\|^2 + 2(\boldsymbol{x}^\top\boldsymbol{y})^2\Big)^n - (\boldsymbol{x}^\top\boldsymbol{y})^{2n}\Big]$
Complex Gaussian	$D^{-1}\Big[\Big(\\|\boldsymbol{x}\\|^2\\|\boldsymbol{y}\\|^2 + (\boldsymbol{x}^\top\boldsymbol{y})^2\Big)^n - (\boldsymbol{x}^\top\boldsymbol{y})^{2n}\Big]$
Real Rademacher	$D^{-1}\Big[\Big(\\|\boldsymbol{x}\\|^2\\|\boldsymbol{y}\\|^2 + 2\Big((\boldsymbol{x}^\top\boldsymbol{y})^2 - \sum_{k=1}^d x_k^2y_k^2\Big)\Big)^n - (\boldsymbol{x}^\top\boldsymbol{y})^{2n}\Big]$
Complex Rademacher	$D^{-1}\Big[\Big(\\|\boldsymbol{x}\\|^2\\|\boldsymbol{y}\\|^2 + (\boldsymbol{x}^\top\boldsymbol{y})^2 - \sum_{k=1}^d x_k^2y_k^2\Big)^n - (\boldsymbol{x}^\top\boldsymbol{y})^{2n}\Big]$
Real TensorSRHT	Real Rademacher Variance $-\frac{c(D,d)}{D^2}\Big[(\boldsymbol{x}^\top\boldsymbol{y})^{2n} - \Big((\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{1}{d-1}\Big(\\|\boldsymbol{x}\\|^2\\|\boldsymbol{y}\\|^2 + (\boldsymbol{x}^\top\boldsymbol{y})^2 - 2\sum_{k=1}^d x_k^2y_k^2\Big)\Big)^n\Big]$
Complex TensorSRHT	Complex Rademacher Variance $-\frac{c(D,d)}{D^2}\Big[(\boldsymbol{x}^\top\boldsymbol{y})^{2n} - \Big((\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{1}{d-1}\Big(\\|\boldsymbol{x}\\|^2\\|\boldsymbol{y}\\|^2 - \sum_{k=1}^d x_k^2y_k^2\Big)\Big)^n\Big]$
Conv. Sur. TensorSRHT	$\begin{cases} D^{-1}\Big(V_q^{(n)} + (d-1)\mathrm{Cov}_q^{(n)}\Big) & \text{if } \mathrm{Cov}_q^{(n)} > 0 \text{ or } D > d, \\ D^{-1}\Big(V_q^{(n)} - \mathrm{Cov}_q^{(n)}\Big) + \mathrm{Cov}_q^{(n)} & \text{otherwise.} \end{cases}$
(Real case: $q = 1$)	$V_q^{(n)} = \Big(\ \\|\boldsymbol{x}\\|^2\ \\|\boldsymbol{y}\\|^2 + ((2q-1)^2+1)((\boldsymbol{x}^\top\boldsymbol{y})^2 - \sum_{k=1}^d x_k^2y_k^2)\Big)^n - (\boldsymbol{x}^\top\boldsymbol{y})^{2n}$
(Complex case: $q = 1/2$)	$\mathrm{Cov}_q^{(n)} = \Big((\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{V_q^{(1)}}{d-1}\Big)^n - (\boldsymbol{x}^\top\boldsymbol{y})^{2n}$

Table 1: Closed-form expressions for the variance $\mathbb{V}\big[\Phi_n(\boldsymbol{x})^\top\overline{\Phi_n(\boldsymbol{y})}\big]$ for different random feature maps $\Phi_n : \mathbb{R}^d \to \mathbb{C}^D$ to approximate polynomial kernel of order $n \in \mathbb{N}$. Here, $D \in \mathbb{N}$ is the number of random features and $c(D,d) := \lfloor D/d \rfloor d(d-1) + (D \mod d)(D \mod d-1)$. See Sections 3 and 4 for details and more generic results. We also show convex surrogate functions in Eq. (70) and Eq. (71) for the variance of TensorSRHT derived in Appendix C.

Note that the integrals in Eq. (46) with respect to P are not available in practice, as P is the unknown data distribution. We instead assume that an i.i.d. sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ of size $m \in \mathbb{N}$ from P is available. This sample may be a subsample of a larger dataset from P. For example, in a supervised learning problem, $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ may be a random subsample of training input points.

Using the i.i.d. sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$, the objective function in Eq. (46) can then be unbiasedly approximated in a U-statistics form as

$$\frac{1}{m(m-1)}\sum_{i\neq j}\mathbb{V}[\hat{k}(\boldsymbol{x}_i,\boldsymbol{x}_j)] + \frac{1}{m(m-1)}\sum_{i\neq j}\Big(k(\boldsymbol{x}_i,\boldsymbol{x}_j) - \mathbb{E}[\hat{k}(\boldsymbol{x}_i,\boldsymbol{x}_j)]\Big)^2$$

$$= \frac{1}{m(m-1)}\sum_{n=1}^{p}\delta[D_n > 0]\ a_n^2\sum_{i\neq j}\mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top\overline{\Phi_n(\boldsymbol{x}_j)}\right] \tag{49}$$

$$+ \frac{1}{m(m-1)}\sum_{i\neq j}\left(k(\boldsymbol{x}_i,\boldsymbol{x}_j) - \sum_{n=0}^{p}\delta[D_n > 0]\ a_n(\boldsymbol{x}_i^\top\boldsymbol{x}_j)^n\right)^2, \tag{50}$$

$$=: g(p, (D_n)_{n=1}^p) \tag{51}$$

where we used Eq. (47) and Eq. (48).

Finally, we formulate our optimization problem. To make the problem tractable, we search for the degree p of the approximate kernel in Eq. (44) from the range $\{p^*_{\min}, p^*_{\min} +$

$1, \dots, p^*_{\max}\}$, where $p^*_{\min}, p^*_{\max} \in \mathbb{N}$ with $p^*_{\min} < p^*_{\max}$ are lower and upper bounds of p selected by the user. We then define our optimization problem as follows:

$$\min_{p, (D_n)_{n=1}^p} g(p, (D_n)_{n=1}^p) \quad \text{subject to} \quad p \in \{p^*_{\min}, p^*_{\min}+1, \dots, p^*_{\max}\}, \tag{52}$$

$$D_n \in \{0, \dots D_{\text{total}}\}, \quad \sum_{n=1}^p D_n = D_{\text{total}}, \quad D_n \geq 1 \text{ if and only if } a_n > 0 \quad (n = 1, \dots, p).$$

where $g(p, (D_n)_{n=1}^p)$ is defined in Eq. (51).

To present our approach to solving Eq. (52), we will first define a simplified optimization problem and describe an algorithm for solving it. We will then use this simplified problem and its solver to develop a solver for the full problem in Eq. (52).

5.3.2 Solving a Simplified Problem

We consider a simplified problem of Eq. (52) in which the polynomial degree $p \in \mathbb{N}$ is fixed and given, and the number of random features D_n is positive, $D_n \geq 1$, for every polynomial degree $n = 1, \dots, p$ with $a_n > 0$. Note that the bias term of the objective function $g(p, (D_n)_{n=1}^n)$, i.e. Eq. (50), only depends on $(D_n)_{n=1}^n$ through the indicator function $\delta[D_n > 0]$. Therefore, under the constraint that $D_n \geq 1$ for all $n = 1, \dots, p$ with $a_n > 0$, Eq. (50) becomes constant with respect to $(D_n)_{n=1}^p$.

Thus, the optimization problem Eq. (52) under the additional constraint of p being fixed and $D_n \geq 1$ for all $n = 1, \dots, p$ with $a_n > 0$ is equivalent to the following optimization problem:

$$\min_{(D_n)_{n=1}^p} \frac{1}{m(m-1)} \sum_{n=1}^p a_n^2 \sum_{i \neq j} \mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right] \quad \text{subject to} \tag{53}$$

$$D_n \subset \{0, \dots D_{\text{total}}\}, \quad \sum_{n=1}^p D_n = D_{\text{total}}, \quad D_n \geq 1 \text{ if and only if } a_n > 0 \quad (n = 1, \dots, p).$$

This is a discrete optimization problem with one equality constraint, and is an instance of the so-called *Resource Allocation Problem* (Floudas and Pardalos, 2009).

We discuss properties of the objective function in Eq. (53) and describe a solver. To this end, we first consider the case where $\Phi_n : \mathbb{R}^d \to \mathbb{C}^{D_n}$ is one of the unstructured polynomial sketches in Section 3; we will later explain its extension to structured sketches from Section 4. In this case, we have $\mathbb{V}\left[\Phi_n(\boldsymbol{x})^\top \overline{\Phi_n(\boldsymbol{y})}\right] = C^{(n)}_{\boldsymbol{x},\boldsymbol{y}}/D_n$ for a constant $C^{(n)}_{\boldsymbol{x},\boldsymbol{y}}$ depending on $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ and the polynomial degree $n \in \mathbb{N}$ but not on D_n, as summarized in Table 1. Therefore,

$$a_n^2 \sum_{i \neq j} \mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right] = \frac{a_n^2}{D_n} \sum_{i \neq j} C^{(n)}_{\boldsymbol{x}_i,\boldsymbol{x}_j} \tag{54}$$

is convex and monotonically decreasing with respect to D_n. From this property, one can use the *Incremental Algorithm* (Floudas and Pardalos, 2009, p. 384) to directly solve the optimization problem (53).

Algorithm 2 describes the Incremental Algorithm for solving the simplified problem in Eq. (53). At every iteration, the algorithm finds $n \in \{1, \dots, p\}$ such that adding one more

Algorithm 2: Incremental Algorithm

```
Result: Optimal solution $D_1, \dots, D_p \geq 1$ to the optimization problem (53).
Input: Dot product kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{n=0}^{\infty} a_n (\boldsymbol{x}^\top \boldsymbol{y})^n$ with $a_n \geq 0$, truncation
order $p \in \mathbb{N}$, the total number of random features $D_{\text{total}} \in \mathbb{N}$ ;
Initialize $D_1 = \cdots = D_p = 1$ and $t = 0$ ;
Let $f(D_1, \dots, D_p) := \sum_{n=1}^{p} \ a_n^2 \sum_{i \neq j} \mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right]$.
while $t < D_{\text{total}}$ do
    Find $j^* = \arg\min_{j \in \{1, \dots, p\}} f(D_1, \dots, D_j + 1, \dots, D_p)$ ;
    $D_{j*} = D_{j*} + 1$ ;
    $t = t + 1$ ;
end
```

feature to the feature map Φ_n (i.e., $D_n = D_n + 1$) decreases the objective function most, and sets $D_n = D_n + 1$. Note again that a closed form expression for $\mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right]$ is available from Table 1.

Time and space complexities. The time and space complexities of Algorithm 2 are $\mathcal{O}(pD_{\text{total}})$ and $\mathcal{O}(p)$, respectively. Note that from Eq. (54), the objective function can be written as

$$f(D_1, \dots, D_p) := \sum_{n=1}^{p} \ a_n^2 \sum_{i \neq j} \mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right] = \sum_{n=1}^{p} \frac{a_n^2}{D_n} \sum_{i \neq j} C_{\boldsymbol{x}_i, \boldsymbol{x}_j}^{(n)}$$

with a_n and $C_{\boldsymbol{x}_i, \boldsymbol{x}_j}^{(n)}$ not depending on the optimizing variable D_n. Therefore, one can precompute the term $\sum_{i \neq j} C_{\boldsymbol{x}_i, \boldsymbol{x}_j}^{(n)}$ for each $n = 1, \dots, p$ before starting the iterations in Algorithm 2, and during the iterations one can use the precomputed values of $\sum_{i \neq j} C_{\boldsymbol{x}_i, \boldsymbol{x}_j}^{(n)}$. Thus, while the complexity of precomputing $\sum_{i \neq j} C_{\boldsymbol{x}_i, \boldsymbol{x}_j}^{(n)}$ is $\mathcal{O}(m^2)$, where m is size of the dataset $\boldsymbol{x}_1, \dots, \boldsymbol{x}_m$ defining the objective function (53), the time and space complexities of Algorithm 2 do not depend on m.

Structured case. We assumed here that Φ_n is one of the unstructured sketches studied in Section 3. This choice of Φ_n makes Eq. (54) convex and monotonically decreasing with respect to D_n, which enables the Incremental Algorithm to solve the optimization problem in Eq. (53).

However, if Φ_n is a structured sketch (i.e., either real or complex TensorSRHT) in Section 4, Eq. (54) is not convex with respect to D_n, and the Incremental Algorithm is not directly applicable. To overcome this problem, when Φ_n is a structured sketch, we propose to use convex surrogate functions in Eq. (70) and Eq. (71) derived in Appendix C to replace $\mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right]$ in the objective function (53), and then apply the Incremental Algorithm. We summarize the concrete form of the convex surrogate function in Table 1. For details, see Appendix C.

```
Algorithm 3: Extended Incremental Algorithm
  Result: Optimal polynomial degree $p^* \in \{p_{\min}, \dots, p_{\max}\}$ and feature cardinalities
          $D^* = (D_1, \dots, D_{p^*}) \in \mathbb{N}^{p^*}$ to the full optimization problem (52).
  Input: Dot product kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{n=0}^{\infty} a_n (\boldsymbol{x}^\top \boldsymbol{y})^n$ with $a_n \geq 0$, upper and
  lower bounds $p_{\min}, p_{\max} \in \mathbb{N}$, the total number of random features $D_{\text{total}} \in \mathbb{N}$ ;
  Set $g^* = \infty$, $p^* = p_{\min}$ and $D^* = \{\}$ ;
  forall $p \in \{p_{\min}, \dots, p_{\max}\}$ do
      Solve Algorithm 2 to obtain $D_1, \dots, D_p$ ;
      Compute $g(p, (D_n)_{n=1}^p)$ in Eq. (51) ;
      If $g(p, (D_n)_{n=1}^p) < g^*$, set $g^* = g(p, (D_n)_{n=1}^p)$, $D^* = (D_n)_{n=1}^p$ and $p^* = p$ ;
  end
```

```
Algorithm 4: Improved Random Maclaurin (RM) Features
  Result: Feature map $\Phi(\boldsymbol{x}) \in \mathbb{C}^{D_{\text{total}}+1}$
  Input: Dot product kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \sum_{n=0}^{\infty} a_n (\boldsymbol{x}^\top \boldsymbol{y})^n$ with $a_n \geq 0$, polynomial
  degree $p^* \in \mathbb{N}$ and feature cardinalities $D_1, \dots, D_{p^*}$ from Algorithm 3 ;
  Initialize $\Phi(\boldsymbol{x}) := [\sqrt{a_0}]$
  forall $n \in \{1, \dots, p^*\}$ do
      Let $\Phi_n(\boldsymbol{x}) \in \mathbb{C}^{D_n}$ be an unbiased polynomial sketch of degree $n$ with $D_n$
        features (see Sections 3 and 4) ;
      Append $\sqrt{a_n}\ \Phi_n(\boldsymbol{x})$ to $\Phi(\boldsymbol{x})$ ;
  end
```

5.3.3 Solving the Full Problem

We now address the full problem in Eq. (52) using Algorithm 2 developed for the simplified problem in Eq. (53). Recall that, by fixing $p \in \{p^*_{\min}, \dots, p^*_{\max}\}$ and constraining $D_n \geq 1$ for all $n = 1, \dots, p$, the full problem in Eq. (52) becomes equivalent to the simplified problem in Eq. (53), which can be solved by Algorithm 2. Thus, we propose to solve the full problem in Eq. (52) by i) first performing Algorithm 2 for each $p \in \{p_{\min}, \dots, p_{\max}\}$, ii) then evaluate each solution $(D_n)_{n=1}^p$ by computing the objective function $g(p, (D_n)_{n=1}^p)$ in Eq. (51), and finally pick up p that gives the smallest objective function value.

Algorithm 3 summarizes the whole procedure for solving the full optimization problem in Eq. (52). Algorithm 3 returns the optimal truncation order $p^* \in \{p_{\min}, \dots, p_{\max}\}$ with the corresponding feature cardinalities $D^* = (D_1, \dots, D_{p^*})$. Given these values, one can construct a feature map as summarized in Algorithm 4. Note that the U-statistics in the empirical objective (51) can be precomputed for all $p_{\min}, \dots, p_{\max}$ *before* running any optimization algorithm. They do not need to be re-evaluated for every execution of Algorithm 2.

5.4 Approximating a Gaussian Kernel

Here we describe how to adapt Algorithm 2 and Algorithm 3 for approximating a Gaussian kernel of the form $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|^2/(2l^2))$ with $l > 0$. By Eq. (39), this Gaussian kernel can be written as

$$k(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2l^2}\right) \exp\left(-\frac{\|\boldsymbol{y}\|^2}{2l^2}\right) \sum_{n=0}^{\infty} a_n (\boldsymbol{x}^\top \boldsymbol{y})^n,$$

where $a_n := 1/(n! l^{2n})$ for $n \in \mathbb{N} \cup \{0\}$. Notice that $\left(-\frac{\|\boldsymbol{x}\|^2}{2l^2}\right)$ and $\left(-\frac{\|\boldsymbol{y}\|^2}{2l^2}\right)$ are scalar values and can be computed for any given input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$.

Thus, the objective function $g(p, (D_n)_{n=1}^p)$ in Eq. (51), which is an empirical approximation of the bias-variance decomposition of the mean squared error in Eq. (46) using an i.i.d. sample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m \overset{i.i.d.}{\sim} P$, can be adapted as

$$\begin{aligned} & g(p, (D_n)_{n=1}^p) \qquad (55) \\ =& \frac{1}{m(m-1)} \sum_{n=1}^{p} \delta[D_n > 0]\ a_n^2 \sum_{i \neq j} \exp\left(-\frac{\|\boldsymbol{x}_i\|^2}{l^2}\right) \exp\left(-\frac{\|\boldsymbol{x}_j\|^2}{l^2}\right) \mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right] \\ +& \frac{1}{m(m-1)} \sum_{i \neq j} \left(k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{n=0}^{p} \delta[D_n > 0]\ a_n \exp\left(-\frac{\|\boldsymbol{x}_i\|^2}{2l^2}\right) \exp\left(-\frac{\|\boldsymbol{x}_j\|^2}{2l^2}\right) (\boldsymbol{x}_i^\top \boldsymbol{x}_j)^n \right)^2 . \end{aligned}$$

Accordingly, the objective function of the simplified problem in Eq. (53) is adapted as

$$f(D_1, \ldots, D_p) := \frac{1}{m(m-1)} \sum_{n=1}^{p} a_n^2 \sum_{i \neq j} \exp\left(-\frac{\|\boldsymbol{x}_i\|^2}{l^2}\right) \exp\left(-\frac{\|\boldsymbol{x}_j\|^2}{l^2}\right) \mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right].$$

By these modifications, Algorithm 2 and Algorithm 3 can be used to obtain the optimal truncation order $p^* \in \{p_{\min}, \ldots, p_{\max}\}$ and the corresponding feature cardinalities $D_1, \ldots, D_{p^*}$. Lastly, Algorithm 4 can be adapted by multiplying the scalar value $\exp\left(-\frac{\|\boldsymbol{x}\|^2}{2l^2}\right)$ to the feature map $\Phi(\boldsymbol{x})$ obtained from Algorithm 4: the new feature map is defined as $\Phi'(\boldsymbol{x}) := \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2l^2}\right) \Phi(\boldsymbol{x})$.

5.5 Numerical Illustration of the Objective Function

To gain an insight about the behavior of Algorithm 3, we provide a numerical illustration of the bias and variance terms in the objective function $g(p, (D_n)_{n=1}^p)$ in Eq. (51) (or its version adapted for the Gaussian kernel in Eq. (55)). To this end, we used the Fashion MNIST dataset (Xiao et al., 2017) and randomly sampled data points $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ with $m = 500$ from the entire dataset of size $60,000$. As a target kernel to approximate, we consider (i) a polynomial kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y}/8 + 7/8)^{20}$ of degree $p = 20$; and (ii) the Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|^2/(2l^2))$, where the length scale $l > 0$ is given by the median heuristic (Garreau et al., 2017), i.e., the median of the pairwise Euclidean distances of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$.

For the polynomial kernel (i), we computed (a) $\frac{a_n^2}{m(m-1)} \sum_{i \neq j} \mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right]$ for each $n = 1, \ldots, p\ (= 20)$, which is the variance component of the objective function in Eq. (51); and (b) $\frac{1}{m(m-1)} \sum_{i \neq j} \left(k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{\nu=0}^{n}\ a_\mu (\boldsymbol{x}_i^\top \boldsymbol{x}_j)^\nu\right)^2$ for each $n = 1, \ldots, p\ (= 20)$,

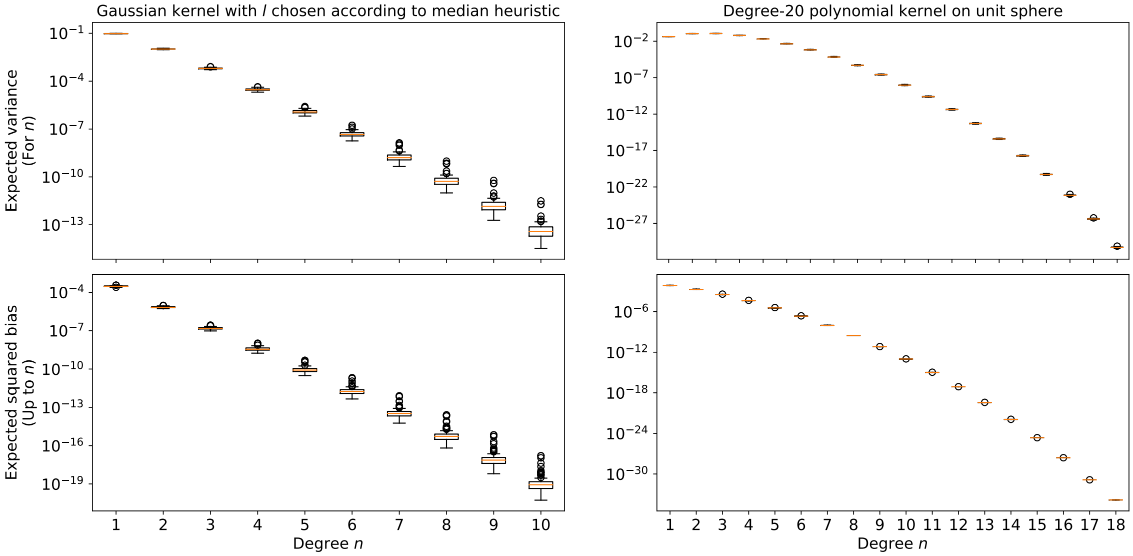

Figure 4: Numerical illustration of Section 5.5. The left two figures are box plots for the Gaussian kernel (i), and the right two figures are those for the polynomial kernel (ii). The top figures show the variance terms (a), and the bottom figures show the bias terms (b). See Section 5.5 for details.

which is the bias component in Eq. (51) computed up to n-th order. For the Gaussian kernel (ii), we computed corresponding quantities from the objective function in Eq. (55): (a) $\frac{a_n^2}{m(m-1)} \sum_{i \neq j} \exp\left(-\frac{\|\boldsymbol{x}_i\|^2}{l^2}\right) \exp\left(-\frac{\|\boldsymbol{x}_j\|^2}{l^2}\right) \mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right]$ for $n = 1, \ldots, 10$ and (b) $\frac{1}{m(m-1)} \sum_{i \neq j} \left(k(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{\nu=0}^{n} \ a_\nu \exp\left(-\frac{\|\boldsymbol{x}_i\|^2}{2l^2}\right) \exp\left(-\frac{\|\boldsymbol{x}_j\|^2}{2l^2}\right) (\boldsymbol{x}_i^\top \boldsymbol{x}_j)^\nu\right)^2$ for $n = 1, \ldots, 10$. We used the real Gaussian sketch for the feature map Φ_n, for which Eq. (20) gives a closed form expression of the variance $\mathbb{V}\left[\Phi_n(\boldsymbol{x}_i)^\top \overline{\Phi_n(\boldsymbol{x}_j)}\right]$; see also Table 1. We set $D_n = 1$ for each n to be evaluated (i.e., $\Phi_n(\boldsymbol{x}) \in \mathbb{R}$.)

To compute the means and standard deviations of the above quantities (a) and (b), we repeated this experiment 100 times by independently subsampling $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ with $m = 500$ from the entire dataset each time. Fig. 4 describes the results. First, we can see that the standard deviations of the quantities (a) and (b) are relatively small, and thus a subsample $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ of size $m = 500$ is sufficient for providing accurate approximations of the respective population quantities of (a) and (b) (where the empirical average with respect to $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ is replaced by the corresponding expectation) in this setting.

Regarding the polynomial kernel (i), the variance terms (a) for polynomial degrees up to $n = 3$ have similar magnitudes, and they decay exponentially fast for polynomial degrees larger than $n = 3$ (notice that the vertical axis of the plot is in log scale). On the other hand, the bias term (b) decays exponentially fast as the polynomial degree n increases. These trends suggest that Algorithm 3 would assign more features to lower order degrees n, in particular to the degree 3 or less. One explanation of these trends is that the parametrization of the kernel $k(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}^\top \boldsymbol{y}/8 + 7/8)^{20}$ gives larger coefficients to lower polynomial degrees in the Maclaurin expansion (see Eq. (37)), and that the distribution

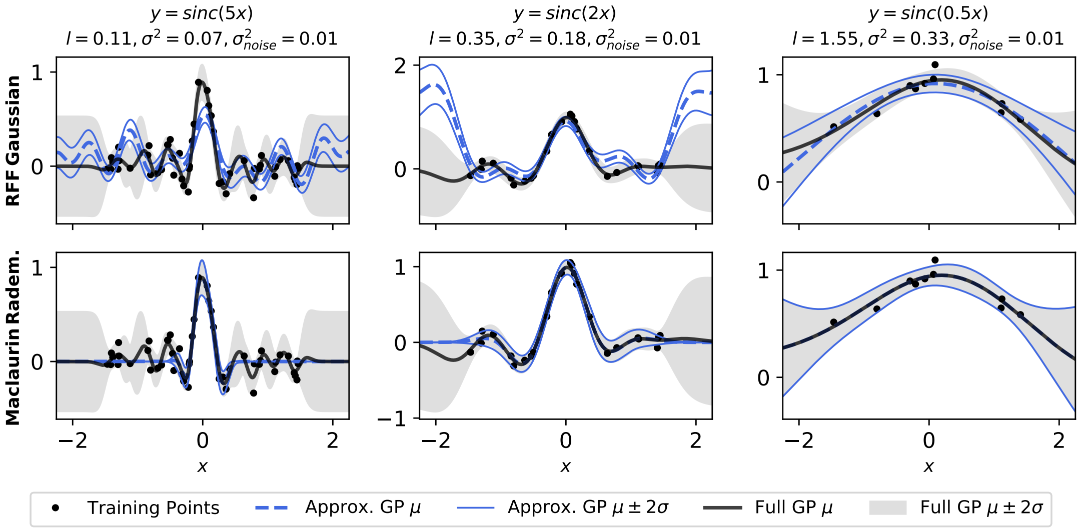

Figure 5: One-dimensional GP regression experiment in Section 5.6. The top row shows the results of random Fourier features (Gaussian RFF), and the bottom row those of the optimized Maclaurin approach. The left, middle, and right columns correspond to the ground-truth sinc functions with frequencies of 5, 2, and 0.5, respectively. The values of l and σ^2 are the kernel hyperparameters obtained by maximizing the log likelihood of training data in the full GP (i.e., without approximation). Dashed black curves represent approximate GP posterior mean functions; black curves represent the posterior means plus and minus 2 times approximate posterior standard deviations; black curves represent the posterior mean functions of the full GP; and the shaded areas are the full GP posterior means plus and minus 2 times the full GP posterior deviations.

of pairwise inner products of the data points $\boldsymbol{x}_1, \dots \boldsymbol{x}_m$ is centered around zero in this experiment.

Regarding the Gaussian kernel (ii), both the variance term (a) and the bias term (b) decay exponentially fast as the polynomial degree n increases. This trend suggests that Algorithm 3 would assign more features to lower order polynomial degrees n.

To summarize, these observations suggest that, to minimize the mean squared error of the approximate kernel, it is more advantageous to assign more features to lower degree polynomial approximations. Algorithm 3 automatically achieves such feature assignments. Additional basic experiments for Algorithm 3 are reported in Appendix F.

5.6 Gaussian Process Regression Toy Example

We performed a toy experiment on one-dimensional Gaussian process (GP) regression, whose results are described in Fig. 5. The purpose is to gain a qualitative understanding of the optimized Maclaurin approximation in Section 5.3 (Algorithm 3). For comparison, we also used Random Fourier Features (RFF) of Rahimi and Recht (2007) in this experiment. We use the real Rademacher sketch in the optimized Maclaurin approach.

We define the ground-truth function as a sinc function, $f(x) = \sin(ax)/x$, with $a > 0$, for which we consider three settings: $a \in \{5, 2, 0.5\}$. We generated training data by adding independent Gaussian noises of variance $\sigma^2_{\text{noise}} = 0.01$ to the ground-truth function $f(x)$.

With this value of noise variance σ^2_{noise}, we then fit a GP regressor using the Gaussian kernel $k(x, y) = \sigma^2 \exp(-(x-y)^2/(2l^2))$ to the training data, where we determined the hyperparameters $l, \sigma^2 > 0$ by maximizing the log marginal likelihood (e.g., Rasmussen and Williams, 2006, Chapter 2). We used the resulting posterior GP as a ground-truth and call it "full GP", treating it as a reference for assessing the quality of approximate GPs. As such, we used the same hyperparameters as the full GP in approximate GPs; this enables evaluating the effects of the approximation in the resulting GP predictive distributions.

We set the number of random features as $D = 10$. In this case, the optimized Maclaurin approach in Algorithm 3 selects the truncation degree $p^* = 9$ and simply allocates the feature cardinalities as $D_1 = \cdots = D_9 = 1$. (Note that one feature is always allocated to the degree $n = 0$). This behavior is because the variance of the Rademacher sketch in Eq. (18) is zero for all polynomial degrees n, as the input dimension is one ($d = 1$) in this experiment.[12]

We can make the following observations from Fig. 5. First, with the optimized Maclaurin approach, the approximate GP posterior mean function approximates the full GP posterior mean function around $x = 0$ more accurately than RFF. Moreover, the range of x on which the Maclaurin approach is accurate becomes wider for a lower frequency a of the ground-truth sinc function (for which the length scale l is larger). This tendency suggests that the Maclaurin approach may be more advantageous than RFF in approximating around $x = 0$ and when the length scale l is relatively large. Experiments in the next section, in particular those with high dimensional datasets, provide further support for this observation.

One issue with the Maclaurin approximation is that, as can be seen from Fig. 5, the approximate GP posterior variance tends to collapse for an input location x far from 0. This behavior may be explained as follows. Recall that in general, the GP posterior variance at location $\boldsymbol{x}$ with an approximate kernel $\hat{k}$ can be written in the form

$$\hat{k}(\boldsymbol{x}, \boldsymbol{x}) - \hat{\ell}_N(\boldsymbol{x}, \boldsymbol{x}), \tag{56}$$

where $\hat{\ell}_N(\boldsymbol{x}, \boldsymbol{x}) \geq 0$ is a data-dependent term (see e.g., Rasmussen and Williams, 2006, Chapter 2). Since $\hat{\ell}_N(\boldsymbol{x}, \boldsymbol{x})$ is non-negative, the GP posterior variance is thus upper-bounded by $\hat{k}(\boldsymbol{x}, \boldsymbol{x})$. Note that the expectation of the Maclaurin-approximate kernel in Eq. (44) for the Gaussian kernel (see also Eq. (39)) is given by

$$\mathbb{E}[\hat{k}(\boldsymbol{x}, \boldsymbol{x})] = \exp\left(-\left\|\frac{\boldsymbol{x}}{l}\right\|^2\right) \cdot \sum_{n=0}^{p} \frac{1}{n!} \left\|\frac{\boldsymbol{x}}{l}\right\|^{2n},$$

which decays to 0 when $\|\boldsymbol{x}/l\|$ is large (because of the finite truncation of the Maclaurin expansion). Therefore, when $\|\boldsymbol{x}/l\|$ is large, the approximate GP posterior variance would decay to 0 accordingly.

One possible (and easy) way of fixing this issue is to add a bias correction term $k(\boldsymbol{x}, \boldsymbol{x}) - \mathbb{E}[\hat{k}(\boldsymbol{x}, \boldsymbol{x})] \geq 0$ to the posterior variance in Eq. (56). In this way, we can prevent the underestimation of the posterior variances with the Maclaurin approach where $\|\boldsymbol{x}/l\|$ is large, which is where the approximate GP posterior mean function may not be accurate and thus preventing the underestimation is desirable.

[12] Thus, the error of the optimized Maclaurin approach stems solely from the finite truncation of the Maclaurin expansion in Eq. (44).

Classification	Num. data points N	Dimensionality d	Regression	Num. data points N	Dimensionality d
Adult	48,842	128	Boston	506	16
Cod_rna	331,152	8	Concrete	1,030	8
Covertype	581,012	64	Energy	768	8
EEG	14,980	16	kin8nm	8,192	8
FashionMNIST	70,000	1,024	Naval	11,934	16
Magic	19,020	16	Powerplant	9,568	4
MNIST	70,000	1,024	Protein	45,730	16
Mocap	78,095	64	Yacht	308	8

Table 2: Datasets used in the experiments. The left and right columns are datasets for classification and regression, respectively.

6. Experiments

In this section, we perform systematic experiments to evaluate the various approaches to approximating dot product kernels discussed in this paper. These approaches include real and complex polynomial sketches in Sections 3 and 4, as well as the optimized Maclaurin approach in Section 5. We consider approximations of both polynomial kernels and Gaussian kernels.

We evaluate the performance of each approximation approach in terms of both i) the accuracy in kernel approximation and ii) the performance in downstream tasks. The downstream tasks we consider are Gaussian process regression and classification. For completeness, we explain how to use complex-valued random features in Gaussian process inference and discuss the resulting computational costs in Appendix D.

In Section 6.1, we first explain the setup of the experiments. In Section 6.2, we describe experiments on polynomial kernel approximation, comparing various approximation approaches. In Section 6.3, we report the results of the wall-clock time comparison of real and complex random features, focusing on the downstream task performance of GP classification. In Section 6.4, we present detailed evaluations of the optimized Maclaurin approach for polynomial and Gaussian kernel approximations in GP classification and regression. Additional experiments are reported in Appendix E.

6.1 Experimental Setup

We explain here the common setup for the experiments in this section.

6.1.1 Datasets

Table 2 shows an overview of the datasets used in the experiments. All the datasets come from the UCI benchmark (Dua and Graff, 2017) except for Cod_rna (Uzilov et al., 2006), FashionMNIST (Xiao et al., 2017), and MNIST (Lecun et al., 1998). We pad input vectors with zeros so that the input dimensionality d becomes a power of two to support Hadamard projections in TensorSRHT. The train/test split is 90/10 and is recomputed for every random seed for the UCI datasets; otherwise it is predefined.

For each dataset, we use its random subsets of size $m = \min(5000, N_{\text{train}})$ and $m_* = \min(5000, N_{\text{test}})$ to define training and test data in an experiment, respectively, where N_{train} and N_{test} are the sizes of the original training and test datasets. Denote by $X_{\text{sub}} =$

$\{\boldsymbol{x}_1, \dots, \boldsymbol{x}_m\}$ and $X_{*,\text{sub}} = \{\boldsymbol{x}_{*,1}, \dots, \boldsymbol{x}_{*,m_*}\}$ those subsets for training and test, respectively. We repeat each experiment 10 times independently using 10 different random seeds, and hence with 10 different subset partitions.

6.1.2 Target Kernels to Approximate

We consider approximation of (i) polynomial kernels and (ii) Gaussian kernels.

(i) Polynomial kernel approximation. We consider a polynomial kernel of the form

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sigma^2 \left(\left(1 - \frac{2}{a^2}\right) + \frac{2}{a^2} \boldsymbol{x}^\top \boldsymbol{y} \right)^p = \sigma^2 \left(1 - \frac{\|\boldsymbol{x} - \boldsymbol{y}\|^2}{a^2}\right)^p \tag{57}$$

with $p \in \mathbb{N}$, $a \geq 2$, $\sigma^2 > 0$, and $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = 1$. We choose this form of polynomial kernels because we use *Spherical Random Features (SRF)* of Pennington et al. (2015) as one of our baselines, and because SRF approximates the polynomial kernel in Eq. (57) defined on the unit sphere of $\mathbb{R}^d$. We follow a similar experimental setup to the one of Pennington et al. (2015), by setting $a = 2$ and $p \in \{3, 7, 10\}$ in Eq. (57). Compared to Pennington et al. (2015) we drop the case $p = 20$ focusing on more realistic cases of smaller p. To make SRF applicable, we unit-normalize the input vectors in each dataset so that they lie on the unit sphere in $\mathbb{R}^d$. In an experiment where we zero-centralize the input vectors, we unit-normalize after applying the zero-centering. We set σ^2 as the variance of the labels of training subset X_{sub}.

(ii) Gaussian kernel approximation. We consider the approximation of the Gaussian kernel $k(\boldsymbol{x}, \boldsymbol{y}) = \sigma^2 \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|^2/(2l^2))$, where we choose the length scale $l > 0$ by the median heuristic (Garreau et al., 2017), i.e., as the median of pairwise Euclidean distances of input vectors in the training subset X_{sub}. We set $\sigma^2 > 0$ as the variance of the labels of X_{sub}.

6.1.3 Error Metrics

We define several error metrics for studying the quality of each approximation approach.

Relative Frobenius norm error. To define this error metric, we need to define some notation. Let $\Phi : \mathbb{R}^d \to \mathbb{C}^D$ be the (either real or complex) feature map of a given approximation method. For test input vectors $\boldsymbol{X}_{*,\text{sub}} = \{\boldsymbol{x}_{*,1}, \dots, \boldsymbol{x}_{*,m_*}\}$, let $\hat{\boldsymbol{K}} \in \mathbb{C}^{m_* \times m_*}$ be the approximate kernel matrix such that $\hat{\boldsymbol{K}}_{i,j} = \Phi(\boldsymbol{x}_i)^\top \overline{\Phi(\boldsymbol{x}_j)}$. Similarly, let $\boldsymbol{K} \in \mathbb{R}^{m_* \times m_*}$ be the exact kernel matrix such that $\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_{*,i}, \boldsymbol{x}_{*,j})$ with k being the target kernel.

We then define the *relative Frobenius norm error* of $\hat{\boldsymbol{K}}$ against $\boldsymbol{K}$ as:

$$\|\boldsymbol{K} - \hat{\boldsymbol{K}}\|_F / \|\boldsymbol{K}\|_F := \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{m} |\boldsymbol{K}_{i,j} - \hat{\boldsymbol{K}}_{i,j}|^2} \Big/ \sqrt{\sum_{i=1}^{m} \sum_{j=1}^{m} \boldsymbol{K}_{i,j}^2}. \tag{58}$$

This error quantifies the quality of the feature map Φ in terms of the resulting approximation accuracy of the kernel matrix. As the target kernel matrix $\boldsymbol{K}$ is real-valued, we discard the imaginary part of $\hat{\boldsymbol{K}}$ if it is complex-valued, unless otherwise specified.

We define other error metrics in terms of two downstream tasks: Gaussian process (GP) regression and classification (see Appendix D for details of these GP tasks).

Kullback-Leibler (KL) divergence. We measure the *KL divergence* between two posterior predictive distributions at test input points: one is that of an approximate GP and the other is that of the exact GP without approximation; see Eq. (91) in Appendix D for details. For GP classification, we measure the KL divergence between the corresponding latent GPs before transformation. Since there are as many GPs as the number of classes, we report the KL divergence averaged over those classes.

Prediction performance. For GP classification, we use the *test error rate* (i.e., the percentage of misclassified examples) for measuring the prediction performance. For GP regression, we report the *normalized mean squared error (norm. MSE)* between the posterior predictive outputs and true outputs, normalized by the variance of the test outputs. Here, we use the full training data of size N_{train} for computing the approximate GP posterior and the full test data of size N_{test} for evaluating the prediction performance.[13]

Mean negative log likelihood (MNLL). We compute the *mean negative log likelihood (MNLL)* of the test data for the approximate GP predictive distribution. MNLL can capture the quality of prediction uncertainties of the approximate GP model (e.g. Rasmussen and Williams, 2006, p. 23). We use the full training and test data for computing the MNLL, as for the prediction performance.

6.1.4 Other Settings

Optimized Maclaurin approach. For the optimized Maclaurin approach in Algorithm 3, we set $p_{\min} = 2$ and $p_{\max} = 10$. We use the training subset $X_{\text{sub}} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\}$ to precompute the U-statistics in Eq. (49) and Eq. (50).

Regularization parameters. We select the regularization parameter in GP classification and regression by a training-validation procedure. That is, we use the 90 % of training data for training and the remaining 10 % for validation, and select the regularization parameter that maximizes the MNLL on the validation set. For GP classification, we choose the regularization parameter from the range $\alpha \in \{10^{-5}, \ldots, 10^{-0}\}$. For GP regression, we choose the noise variance from the range $\sigma^2_{\text{noise}} \in \{2^{-15}, \ldots, 2^{15}\}$. See Appendix D for the definition of these parameters.

Importantly, we perform this selection procedure using a baseline approach,[14] and after selecting the regularization parameter, we set the *same* regularization parameter for all the approaches (including our optimized Maclaurin approach) for computing error metrics. In this way, we make sure that the selected regularization parameter is not in favour of our approaches (and in this sense we give an advantage to the baseline).

[13] We did not use the full training and test datasets for evaluating the KL divergence, since it requires computing the exact GP posterior on the full training data of size N_{train}, which costs $\mathcal{O}(N^3_{\text{train}})$ and is not feasible for datasets with large N_{train}.

[14] More specifically, we use the Spherical Random Features (SRF) (Pennington et al., 2015) when the target kernel is a polynomial kernel, and Random Fourier Features (Rahimi and Recht, 2007) when the target kernel is Gaussian, for selecting the regularization parameter.

6.2 Polynomial Kernel Approximation

We first study the approximation of the polynomial kernels in Eq. (57), comparing different polynomial sketches in terms of the relative Frobenius norm error in Eq. (58) on Fashion-MNIST. Fig. 6 describes the results. We consider the following polynomial sketches in this experiment:

(i) Gaussian and Rademacher sketches (Section 3). We use the real Gaussian and Rademacher sketches, i.e., the unstructured polynomial sketches in Eq. (5) with Gaussian and Rademacher weights (“Gaussian” and “Rademacher”, respectively, in Fig. 6), along with their complex counterparts (“Gaussian Comp.” and “Rademacher Comp.” in Fig. 6).

(ii) TensorSRHT (Section 4). We consider the real TensorSRHT in Eq. (27) with Rademacher weights (“TensorSRHT” in Fig. 6), and the complex TensorSRHT in Eq. (32) with complex Rademacher weights (“TensorSRHT Comp.” in the figure); see also Algorithm 1.

(iii) Random Maclaurin (Section 5). We use the Random Maclaurin approach explained in Section 5.2. To improve its performance, we truncate the support of the importance sampling measure $\mu(n) = 2^{-(n+1)}$ in Eq. (40) to degrees $n \in \{1, \ldots, p\}$.[15] Note that the term $n = 0$ in Eq. (36) associated with coefficient a_0 does not need to be approximated, as we append $\sqrt{a_0}$ to the feature map. We consider the Random Maclaurin approach using the real Rademacher sketch (“Rnd. Macl. Radem.” in Fig. 6).

(iv) Optimized Maclaurin (Section 5). We consider the optimized Maclaurin approach in Section 5.3 using the Rademacher approach (“Opt. Macl. Radem.” in Fig. 6). We also include the real and complex versions involving TensorSRHT (“Opt. Macl. TensorSRHT” and “Opt. Macl. TensorSRHT Comp.”, respectively, in Fig. 6).

(v) TensorSketch For completeness, we also include in this experiment *TensorSketch* of Pham and Pagh (2013), a state-of-the-art polynomial sketch (“TensorSketch” in Fig. 6).

Setting. We perform the experiments using FashionMNIST (“Non-centered data” in Fig. 6) and its centered version for which we subtract the mean of the input vectors from each input vector (“Centered data” in Fig. 6). For each approach, the number of random features is $D \in \{d, 3d, 5d\}$, where $d = 1,024$ for FashionMNIST.

From the results in Fig. 6, we can make the following observations.

Effectiveness of the optimization approach. The optimized Maclaurin approach with the Rademacher sketch (“Opt. Macl. Radem.”) achieves smaller errors than the corresponding random Maclaurin approach (“Rnd. Macl. Rad.”) for all cases, and with a large margin for $p = 7$ and $p = 10$. This improvement demonstrates the effectiveness of the proposed optimization approach that allocates more features to polynomial degrees with larger variance reduction.

[15] Without this restriction of the support, the randomized Maclaurin approach may sample polynomial degrees n such that $n > p$ from $\mu(n)$, for which the associated coefficient in the Maclaurin expansion in Eq. (37) is zero. Therefore, the resulting feature maps may contain zeros, which are redundant and make the kernel approximation inefficient.

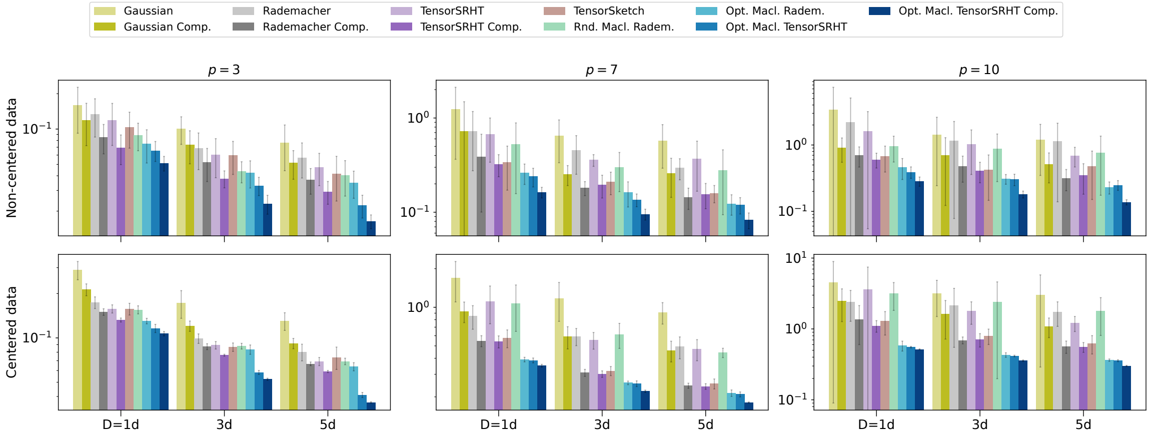

Figure 6: Results of the experiments in Section 6.2 using FashionMNIST. Each plot shows the relative Frobenius norm errors in Eq. (58) of different sketches for approximating the polynomial kernel in Eq. (57) with $p \in \{3, 7, 10\}$ and $D \in \{1d, 3d, 5d\}$. The top and bottom rows show results without and with zero-centring the data, respectively.

Variance reduction by complex features. Complex TensorSRHT achieves significantly smaller errors than the real TensorSRHT ("TensorSRHT"), in particular for small polynomial degrees p. These improvements show the effectiveness of complex features in variance reduction, corroborating the preliminary results shown in Figures 2 and 3. The optimized Maclaurin approach using complex features ("Opt. Macl. TensorSRHT Comp.") also achieves smaller errors than the optimized Maclaurin approach using real features ("Opt. Macl. TensorSRHT") and is quite significant across all methods.

Effectiveness of complex features on non-negative data. The improvements by complex features are more significant for the non-centered data than those for the centered-data. The non-centered data here consist of *non-negative* input vectors, as FashionMNIST consists of such vectors. This observation agrees with the discussion in Section 3.3 suggesting that complex features yield an approximate kernel whose variance is smaller than that of real features, if the input vectors are non-negative.

TensorSRHT v.s. TensorSketch. While the real TensorSRHT produces larger errors than TensorSketch for all the cases except $p = 3$, the complex TensorSRHT outperforms TensorSketch for all the cases. This comparison shows that the use of complex features can make TensorSRHT competitive to the state-of-the-art (and one can further improve its performance by using it in the optimized Maclaurin approach).

6.3 Wall-Clock Time Comparison of Real and Complex Random Features in GP Classification

We consider GP classification using the polynomial kernel in Eq. (57), and compare the approximation quality of real and complex random features, in terms of both the number of features and wall-clock time. As explained in Appendix D.3, the cost of computing an approximate GP posterior using D complex random features is higher than that using D

real features.[16] Therefore, to evaluate the relevance of complex features in practice, we investigate here the approximation quality of complex random features and that of real random features in GP classification, when both are given the *same* computational budget (in wall-clock time).

Setting. We use the Rademacher sketch and TensorSRHT, and their respective complex versions. For each polynomial sketch, we compute the KL divergence (91) between the approximate and exact GP posteriors (see Appendix D for details), and record wall-clock time (in seconds) spent on constructing random features and on computing the approximate GP posterior.[17] We use FashionMNIST for this experiment.

Results. Fig. 7 describes the results. The approximate GPs using complex random features achieve equal or lower KL-divergences than those using real features of the same computation time, for all the cases. In particular, the improvements of complex features are larger for higher polynomial degrees p and for the non-centered (and thus non-negative) data. These observations agree with the corresponding observation in Section 6.2 and the discussion in Section 3.3 on when complex features yield lower variances than real features.

6.4 Systematic Evaluation of the Optimized Maclaurin Approach

Lastly, we systematically evaluate the performance of the optimized Maclaurin approach in Section 5. We run experiments on approximate GP classification and regression on a variety of datasets, using a high-degree polynomial kernel and the Gaussian kernel.

Optimized Maclaurin approach. We consider the optimized Maclaurin approach in Section 5.3. Similarly to the previous experiments, we compare the optimized version for the Rademacher sketch ("Opt. Macl. Radem.") and the real and complex versions of the optimized Maclaurin method using TensorSRHT ("Opt. Macl. TensorSRHT" and "Opt. Macl. TensorSRHT Comp.").

Baselines. We use here approximation approaches based on Random Fourier Features (RFF) (Rahimi and Recht, 2007) and their extensions such as *Spherical Random Features* (SRF) (Pennington et al., 2015) and *Structured Orthogonal Random Features* (SORF) (Yu et al., 2016) as baselines. The latter two approaches constitute the state-of-the-art.

These approaches generate a set of frequency samples $\omega_1, \dots, \omega_{D/2} \in \mathbb{R}^d$ (suppose D is even for simplicity) from a certain spectral density, and construct a feature map[18] of

[16] Specifically, if one uses D complex features, then the inversion of the matrix in Eq. (88) requires 4 times as many floating point operations as the case of using D real features. Note that, if one instead uses $2D$ real features, then the inversion of the matrix in Eq. (88) requires 8 times as many operations as the case of using D real features. Thus, doubling the number of real features is 2 times more expensive than using complex features. See Appendix D.3 for details.

[17] We recorded the time measurements on an NVIDIA P100 GPU and PyTorch version 1.10 with native complex linear algebra support.

[18] There is another popular version of the feature map in Eq. (59) defined as $\Phi_{\mathcal{R}}(\boldsymbol{x}) = \sqrt{\frac{2}{D}} \left[\cos(\boldsymbol{w}_1^\top \boldsymbol{x} + b_1), \dots, \cos(\boldsymbol{w}_D^\top \boldsymbol{x} + b_D)\right]^\top \in \mathbb{R}^D$ with $b_1, \dots, b_D$ uniformly sampled on $[0, 2\pi]$. Following Sutherland and Schneider (2015) who suggested the superiority of Eq. (59), we use Eq. (59) here in all the methods using RFF, including SRF and SORF.

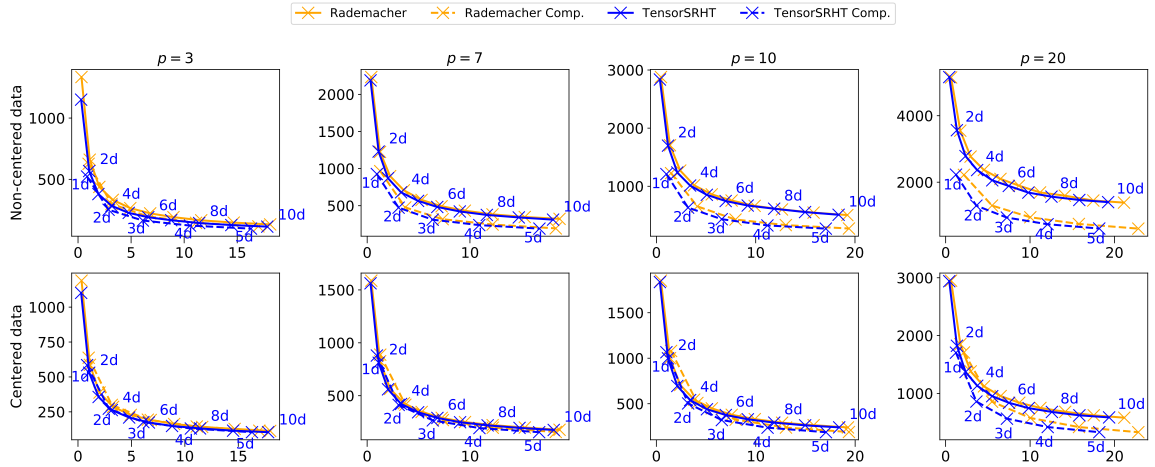

Figure 7: Results of the experiments in Section 6.3 on wall-clock time comparison of real and complex random features in GP classification on FashionMNIST. In each plot, the vertical axis shows the KL divergence (91) between the approximate and the exact GP posteriors for each polynomial sketch, and the horizontal axis is wall-clock time (in seconds) spent on constructing random features and on computing the approximate GP posterior. Each column corresponds to a different degree $p \in \{3, 7, 10, 20\}$ of the polynomial kernel in Eq. (57). The top row shows results on the non-centered (thus non-negative) data, and the bottom row to those on the zero-centered data. The number of random features is $D \in \{1d, \dots, 10d\}$ for real features, and $D \in \{1d, \dots, 5d\}$ for complex features, as annotated next to the respective measurements in each plot.

dimension D as, for any $\boldsymbol{x} \in \mathbb{R}^d$,

$$\Phi_{\mathcal{R}}(\boldsymbol{x}) = \sqrt{\frac{2}{D}} \left[\cos(\boldsymbol{w}_1^\top \boldsymbol{x}), \dots, \cos(\boldsymbol{w}_{D/2}^\top \boldsymbol{x}), \sin(\boldsymbol{w}_1^\top \boldsymbol{x}), \dots, \sin(\boldsymbol{w}_{D/2}^\top \boldsymbol{x})\right]^\top \in \mathbb{R}^D. \tag{59}$$

Each approach has its own way of generating the frequency samples $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_{D/2}$: the original RFF generates them in an i.i.d. manner from the spectral density of a kernel, SORF uses structured orthogonal matrices (thus we may call it “RFF Orth.”), and SRF uses a certain optimized spectral density.

For a thorough comparison, we also consider a complex version of these RFF-based approaches. By generating frequency samples $\boldsymbol{\omega}_1, \dots, \boldsymbol{\omega}_D \in \mathbb{R}^d$ in the specific way of each approach, one can define a corresponding complex feature map as, for any $\boldsymbol{x} \in \mathbb{R}^d$,

$$\Phi_{\mathcal{C}}(\boldsymbol{x}) := \sqrt{\frac{1}{D}} \left[\exp(\mathrm{i}\boldsymbol{\omega}_1^\top \boldsymbol{x}), \dots, \exp(\mathrm{i}\boldsymbol{\omega}_D^\top \boldsymbol{x})\right]^\top \in \mathbb{C}^D. \tag{60}$$

One can see[19] that Eq. (60) is a complex version of Eq. (59) by defining an approximate kernel with $\Phi_{\mathcal{C}}(\boldsymbol{x})$ and taking its real part, which recovers Eq. (59) of dimension $2D$.

[19] Define an approximate kernel with Eq. (60) as $\hat{k}(\boldsymbol{x}, \boldsymbol{y}) := \Phi_{\mathcal{C}}(\boldsymbol{x})^\top \overline{\Phi_{\mathcal{C}}(\boldsymbol{y})} = \frac{1}{D}\sum_{i=1}^D \exp(\mathrm{i}\boldsymbol{\omega}_i^\top(\boldsymbol{x} - \boldsymbol{y})) = \frac{1}{D}\sum_{i=1}^D \exp(\mathrm{i}\boldsymbol{\omega}_i^\top \boldsymbol{x})\overline{\exp(\mathrm{i}\boldsymbol{\omega}_i^\top \boldsymbol{y})}$. By taking its real part, we have $\mathcal{R}\{\hat{k}(\boldsymbol{x}, \boldsymbol{y})\} = \frac{1}{D}\sum_{i=1}^D \cos(\boldsymbol{w}_i^\top(\boldsymbol{x} - \boldsymbol{y})) = \frac{1}{D}\sum_{i=1}^D \left(\cos(\boldsymbol{w}_i^\top \boldsymbol{x})\cos(\boldsymbol{w}_i^\top \boldsymbol{y}) + \sin(\boldsymbol{w}_i^\top \boldsymbol{x})\sin(\boldsymbol{w}_i^\top \boldsymbol{y})\right) =: \Phi_{\mathcal{R}}(\boldsymbol{x})^\top \Phi_{\mathcal{R}}(\boldsymbol{y})$, where $\Phi_{\mathcal{R}}(\boldsymbol{x}) := \sqrt{\frac{1}{D}}\left[\cos(\boldsymbol{w}_1^\top \boldsymbol{x}), \dots, \cos(\boldsymbol{w}_D^\top \boldsymbol{x}), \sin(\boldsymbol{w}_1^\top \boldsymbol{x}), \dots, \sin(\boldsymbol{w}_D^\top \boldsymbol{x})\right]^\top \in \mathbb{R}^{2D}$ is the $2D$-dim. version of Eq. (59).

6.4.1 Approximate GP Inference with Polynomial Kernels

We first consider approximate GP classification and regression with polynomial kernels.

Setting. We set the polynomial degree to $p = 3$ and $p = 7$ to test various approaches on low and moderate degrees. We apply zero-centering to each dataset (i.e., we subtract the mean of input vectors from each input vector), as it improves the MNLL values on most datasets (see Appendix E for supplementary experiments). We evaluate all the four error metrics in Section 6.1.3, including the relative Frobenius norm error in Eq. (58). For each approach, the number of random features is $D \in \{d, 3d, 5d\}$ with d being the dimensionality of input vectors.

Baselines. As a baseline, we use SRF (Pennington et al., 2015), a state-of-the-art approach to approximating polynomial kernels defined on the *unit sphere* in $\mathbb{R}^d$. Pennington et al. (2015) show that SRF works particularly well for approximating high degree polynomial kernels, and significantly outperforms the Random Maclaurin approach (Kar and Karnick, 2012) and TensorSketch (Pham and Pagh, 2013) for such kernels.

We also consider two other extensions of SRF for baselines. SRF generates the frequency samples $\boldsymbol{\omega}_1, \ldots, \boldsymbol{\omega}_{D/2}$ in Eq. (59) from an optimized spectral density, by first drawing samples from the unit sphere in $\mathbb{R}^d$. Therefore, by replacing these samples on the unit sphere by structured orthogonal projections of SORF (Yu et al., 2016), one can construct a structured version of SRF. We use this structured SRF as another baseline ("SRF Orth."). Moreover, we consider a complex extension of the structured SRF in the form of Eq. (60) ("SRF Orth. Comp."). While these extensions are themselves novel, we include them in the experiments, as they improve over the vanilla SRF and make the experiments more competitive. Finally, we also include the method in Ahle et al. (2020) and its complex version.

Fig. 8 and Fig. 9 show the results of approximate GP classification on four datasets from Table 2. We present the results on the other four datasets as well as the results of GP regression in Appendix E to save space. We can make the following observations from these results.

Relative Frobenius norm error. For most cases, the optimized Maclaurin approaches with TensorSRHT achieve lower relative Frobenius norm errors than the SRF approaches and the method by Ahle et al. (2020). Across all methods, there are cases where the improvements offered by the complex approach are quite large.

KL divergence. While the optimized Maclaurin approaches achieve lower KL divergences than the SRF approaches for most cases, the margins are smaller than those for the relative Frobenius norm errors. One possible reason is that the Maclaurin approaches in general (either random or optimized) can be inaccurate in approximating the GP posterior variances at test inputs far from $\boldsymbol{x} = \mathbf{0}$, as discussed in Section 5.6. While we suggested a way of fixing this issue in Section 5.6, we do not implement it to conduct a direct comparison with the SRF approaches.

Classification errors and mean negative log likelihood (MNLL). The optimized Maclaurin approaches with TensorSRHT achieve equal or lower classification errors and MNLL than the SRF approaches. These results suggest that the optimized Maclaurin

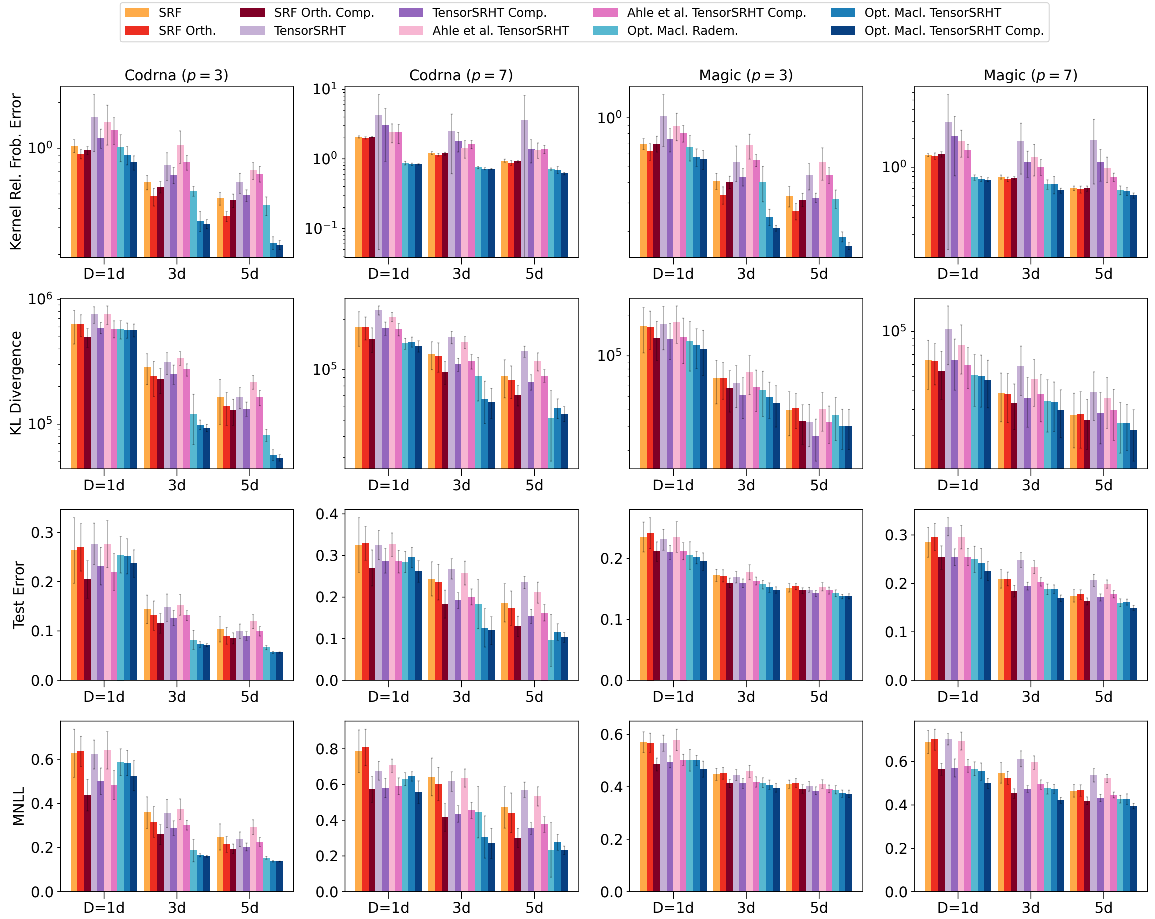

Figure 8: Codrna and Magic results of the experiments in Section 6.4.1 on approximate GP classification with polynomial kernels of degree $p = 3$ and $p = 7$. Lower values are better for all the metrics. For each dataset, we show the number of random features $D \in \{1d, 3d, 5d\}$ used in each method on the horizontal axis, with d being the input dimensionality of the dataset.

approaches are promising not only in kernel approximation accuracy but also in downstream task performance. Recall that we selected the regularization parameter in GP classification by maximizing the MNLL of SRF (on the validation set), and used the same regularization parameter in the other approaches (See Section 6.1.4). Therefore, the results of Fig. 8 and Fig. 9 are in favor of the SRF approaches, and the optimized Maclaurin approaches may perform even better if we choose the regularization parameter for them separately.

6.4.2 Approximate GP Inference with a Gaussian kernel

We next consider GP classification using a Gaussian kernel. As in Section 6.4.1, we apply zero-centring to the input vectors of each dataset.

Baselines. We use RFF, SORF ("RFF Orth.") and a complex extension of SORF ("RFF Orth. Comp.") as baselines (see the beginning of Section 6.4 for details). SORF is a state-of-

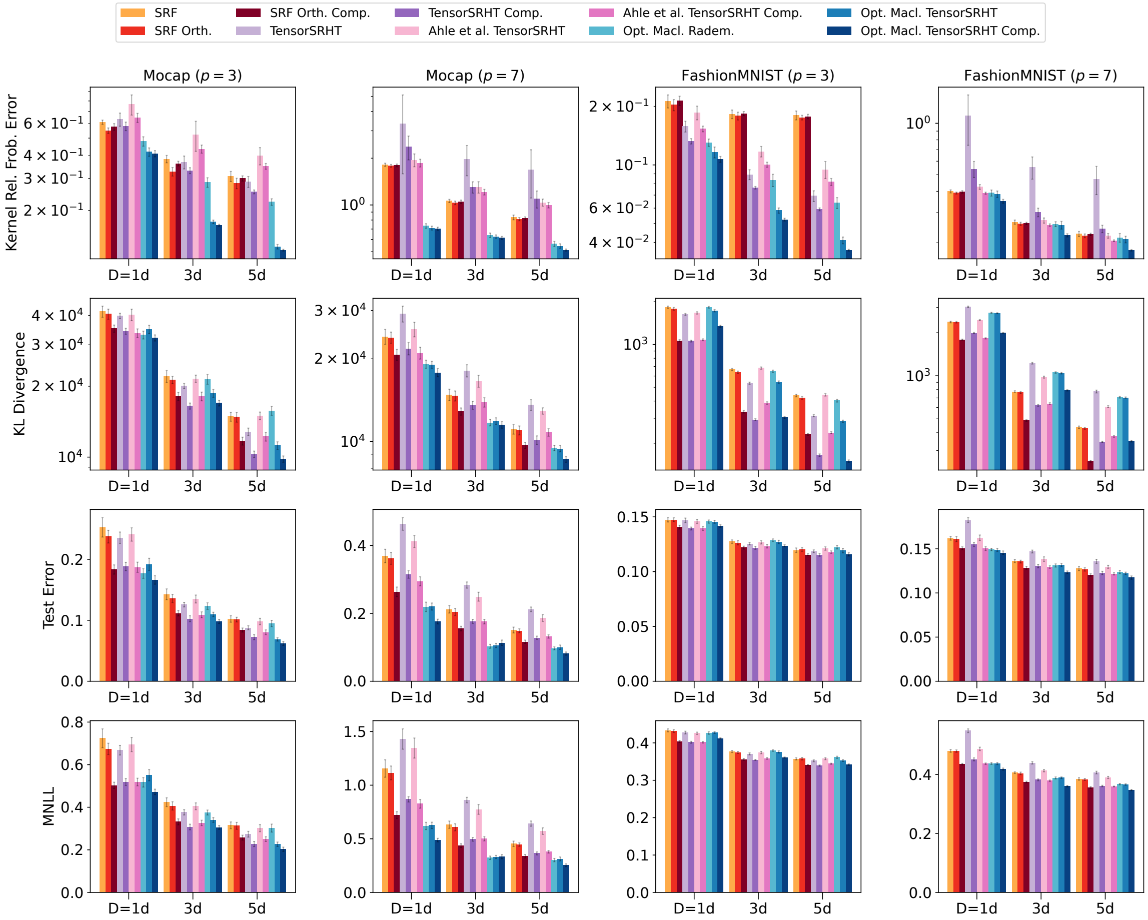

Figure 9: Mocap and FashionMNIST results of the experiments in Section 6.4.1 on approximate GP classification with polynomial kernels of degree $p = 3$ and $p = 7$. Lower values are better for all the metrics. For each dataset, we show the number of random features $D \in \{1d, 3d, 5d\}$ used in each method on the horizontal axis, with d being the input dimensionality of the dataset.

the-art approach to approximating a Gaussian kernel (e.g. Choromanski et al., 2018). As in Section 6.4.1, we consider its complex extension to make the experiments more competitive.

Results. Fig. 10 summarizes the results on four datasets from Table 2. We show the results on the rest of the datasets as well as the results of GP regression in Appendix E. We can make similar observations for Fig. 10 as for the polynomial kernel experiments in Section 6.4.1 (and thus we omit explaining them). The results suggest the effectiveness of the optimized Maclaurin approach with TensorSRHT in approximating the Gaussian kernel.

6.4.3 Influence of the Data Distribution on the Kernel Approximation

Lastly, we investigate a characterization of datasets for which the optimized Maclaurin approach performs well. We focus on polynomial kernel approximation, and make a comparison with SRF as in Section 6.4.1.

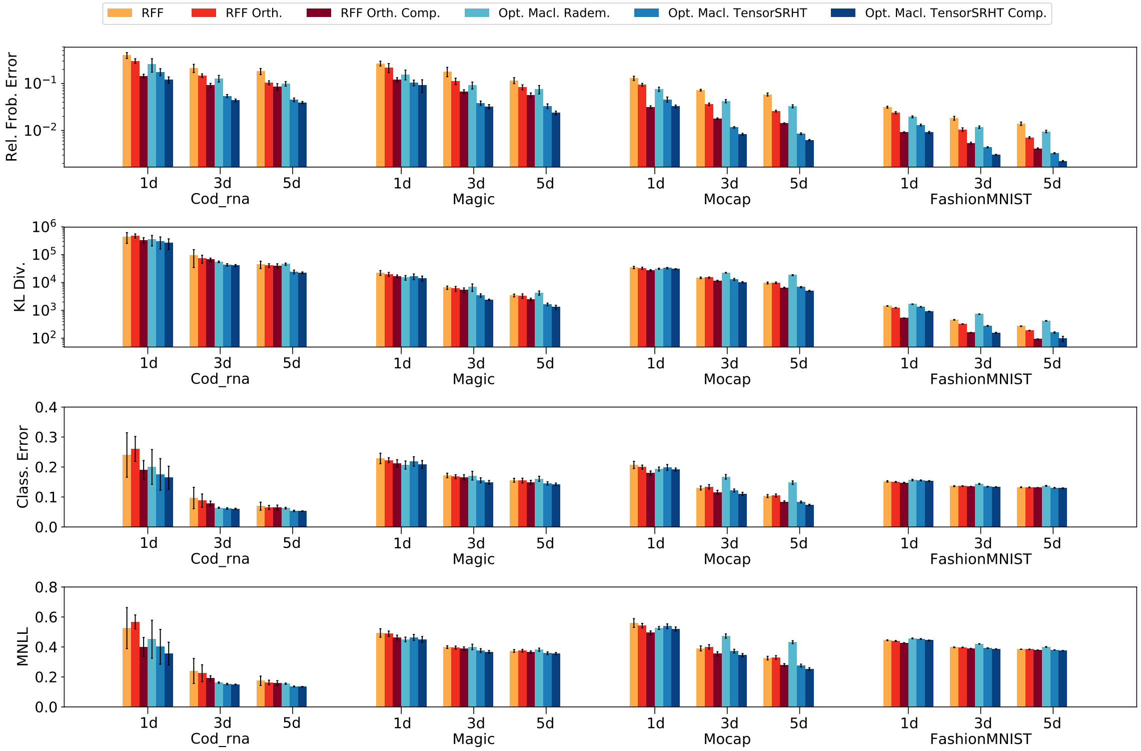

Figure 10: Results of the experiments in Section 6.4.2 on approximate GP classification with a Gaussian kernel. Lower values are better for all the metrics. For each dataset, we show the number of random features $D \in \{1d, 3d, 5d\}$ used in each method on the horizontal axis, with d being the input dimensionality of the dataset. We put the legend labels and the bars in the same order.

Fig. 11 describes a histogram of pairwise distances $\{\|\boldsymbol{x}_{*,i} - \boldsymbol{x}_{*,j}\|\}_{i \neq j}$ of the input vectors in a test subset $X_{*,\text{sub}} = \{\boldsymbol{x}_{*,1}, \ldots, \boldsymbol{x}_{*,m_*}\}$, obtained after zero-centering and unit-normalization, of each of four representative datasets (kin8nm, Cod_rna, Naval, and Protein). For these datasets, the optimized Macluarin approach and SRF show stark contrasts in their performances; see Section § 6.4.1 and Appendix E. Note that the polynomial kernel in Eq. (57) is a shift-invariant kernel on the unit sphere of $\mathbb{R}^d$, and thus its value depends only on the distance $\tau := \|\boldsymbol{x} - \boldsymbol{y}\|$ between the input vectors $\boldsymbol{x}, \boldsymbol{y}$ as long as $\|\boldsymbol{x}\| = \|\boldsymbol{y}\| = 1$. This motivates us to study here the distribution of pairwise distances and its effects on approximating the polynomial kernel in Eq. (57).

In Fig. 11, the optimized Maclaurin approach yields lower relative Frobenius norm errors (58) than SRF for the left two plots, while the optimized Maclaurin approach is less accurate than SRF for the right two plots. For the datasets of the right two plots (Naval and Protein), the pairwise distances $\{\|\boldsymbol{x}_{*,i} - \boldsymbol{x}_{*,j}\|\}_{i \neq j}$ concentrate around $\tau = 0$ (and there is a smaller mass around $\tau = 2$). In comparison, for the datasets of the left two plots (kin8nm and Cod_rna), the pairwise distances are relatively more evenly distributed across the possible range $\tau \in [0, 2]$.

The above observation suggests that the optimized Maclaurin approach is more suitable for datasets in which the pairwise distances $\{\|\boldsymbol{x}_{*,i} - \boldsymbol{x}_{*,j}\|\}_{i \neq j}$ are not concentrating around

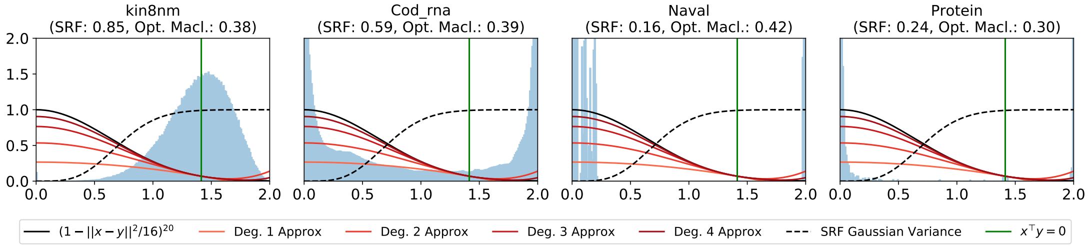

Figure 11: Histograms of pairwise Euclidean distances $\{\|\boldsymbol{x}_{*,i} - \boldsymbol{x}_{*,j}\|\}_{i \neq j}$ for test subsets of four datasets (Section 6.4.3). On the top of each figure, we show the relative Frobenius norm errors (58) of the optimized Maclaurin approach with real TensorSRHT and of SRF with structured orthogonal projections. The black curve represents the polynomial kernel in Eq. (57) with $p = 20$ as a function of $\tau := \|\boldsymbol{x} - \boldsymbol{y}\|$ (the horizontal axis); the orange curves describe its degree $n \in \{1, 2, 3, 4\}$ approximations (i.e., the truncation of the Maclaurin expansion (37) of the polynomial kernel up to the n-th degree terms.). The dashed curve represents the variance of the SRF approximation as a function of $\tau = \|\boldsymbol{x} - \boldsymbol{y}\|$. The green vertical line shows the value of $\tau = \|\boldsymbol{x} - \boldsymbol{y}\| = \sqrt{2}$ for which the input vectors $\boldsymbol{x}, \boldsymbol{y}$ are orthogonal, $\boldsymbol{x}^\top \boldsymbol{y} = 0$.

0, i.e., datasets in which there is a diversity in the input vectors $\{\boldsymbol{x}_{*,1}, \dots, \boldsymbol{x}_{*,m_*}\}$. In fact, for approximating the polynomial kernel (black curve in Fig. 11), the finite-degree Maclaurin approximations (orange curves) tend to be less accurate for input vectors $\boldsymbol{x}, \boldsymbol{y}$ close to each other, $\tau = \|\boldsymbol{x} - \boldsymbol{y}\| \approx 0$, and become relatively more accurate as input vectors $\boldsymbol{x}, \boldsymbol{y}$ approach orthogonality, i.e. $\boldsymbol{x}^\top \boldsymbol{y} = 0$ (or $\tau = \|\boldsymbol{x} - \boldsymbol{y}\| = \sqrt{2}$; the vertical green line); see also the Maclaurin expansion (37) of the polynomial kernel. On the other hand, the variance of SRF is the lowest around $\tau = \|\boldsymbol{x} - \boldsymbol{y}\| = 0$ and increases as τ tends to 2. Therefore, the SRF performs well if the pairwise distances $\{\|\boldsymbol{x}_{*,i} - \boldsymbol{x}_{*,j}\|\}_{i \neq j}$ concentrate around 0, and may become inaccurate if they do not.

7. Conclusion

We made several contributions for understanding and improving random feature approximations for dot product kernels. First, we studied polynomial sketches, i.e., random features for polynomial kernels, such as the Rademacher sketch and TensorSRHT, and discussed their generalizations using complex-valued features. We derived closed form expressions for the variances of these polynomial sketches, which are useful in both theory and practice.

On the theoretical side, these variance formulas provide novel insights into these polynomial sketches, such as conditions for a structured sketch to have a lower variance than the corresponding unstructured sketch, and conditions for a complex sketch to have a lower variance than the corresponding real sketch. Our systematic experiments support these findings. On the practical side, these variance formulas can be evaluated in practice, and therefore enable us to estimate the mean squared errors of the approximate kernel for given input points.

Based on the derived variance formulas, we developed a novel optimization algorithm for data-driven random feature approximations of dot product kernels, which is also applicable to the Gaussian kernel. This approach uses a finite Maclaurin approximation of the kernel, which approximates the kernel as a finite sum of polynomial kernels of different degrees.

Given a total number of random features, our optimization algorithm determines how many random features should be used for each polynomial degree in the Maclaurin approximation. We defined the objective function of this optimization algorithm as an estimate of the averaged mean squared error regarding the data distribution, and used the variance formulas for this purpose. We empirically demonstrated that this optimized Maclaurin approach achieves state-of-the-art performance on a variety of datasets, both in terms of the kernel approximation accuracy and downstream task performance.

As described in the introduction, dot product kernels have been actively used in many domains of applications, such as genomic data analysis, recommender systems, computer vision, and natural language processing. In these applications, interactions among input variables have significant effects on the output variables of interest, and thus dot product kernels offer an appropriate modeling tool. In particular, dot product kernels are being used in an inner-loop of larger neural network models, such as the dot product attention mechanism used in Transformer architectures (Vaswani et al., 2017; Choromanski et al., 2021).

One major challenge of using dot product kernels is the computational efficiency, and random feature approximations offer a promising solution. Our contributions improve the efficiency of random feature approximations, and we hope that these contributions make dot product kernels even more useful in the above application domains.

Acknowledgments

The Authors wish to thank the Associate Editor and the anonymous Reviewers for their insightful reviews and comments. This work has been in part supported by the French government through the 3IA Cote d'Azur Investment in the Future Project, which is managed by the National Research Agency (ANR) and has the reference number ANR-19-P3IA-0002. MF also gratefully acknowledges support from the AXA Research Fund and ANR (grant ANR-18-CE46-0002).

Appendix A. Proofs for Section 3

A.1 Proof of Theorem 2

We first show

$$\begin{aligned}\mathbb{V}[\hat{k}_{\mathcal{C}}(\boldsymbol{x},\boldsymbol{y})] = \Bigg(& \sum_{k=1}^{d} \mathbb{E}[|z_k|^4] x_k^2 y_k^2 + \|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2 - 2\sum_{k=1}^{d} x_k^2 y_k^2 + (\boldsymbol{x}^\top \boldsymbol{y})^2 \\ & + \sum_{i=1}^{d} \sum_{\substack{j=1 \\ j\neq i}}^{d} \mathbb{E}[z_i^2]\mathbb{E}[\overline{z_j}^2] x_i x_j y_i y_j \Bigg)^p - (\boldsymbol{x}^\top \boldsymbol{y})^{2p}. \qquad (61)\end{aligned}$$

where $z_i^2 := z_i z_i$ and $\overline{z_i}^2 := \overline{z_i z_i}$ are in general different from $|z_i|^2 = z_i \overline{z_i}$. We have

$$\begin{aligned}\mathbb{V}[\hat{k}_{\mathcal{C}}(\boldsymbol{x},\boldsymbol{y})] &= \mathbb{E}[|\hat{k}_{\mathcal{C}}(\boldsymbol{x},\boldsymbol{y})|^2] - |\mathbb{E}[\hat{k}_{\mathcal{C}}(\boldsymbol{x},\boldsymbol{y})]|^2 = \mathbb{E}[|\prod_{i=1}^{p} \boldsymbol{z}_i^\top \boldsymbol{x} \overline{\boldsymbol{z}_i^\top \boldsymbol{y}}|^2] - (\boldsymbol{x}^\top \boldsymbol{y})^{2p} \\ &= \prod_{i=1}^{p} \mathbb{E}[|\boldsymbol{z}_i^\top \boldsymbol{x} \overline{\boldsymbol{z}_i^\top \boldsymbol{y}}|^2] - (\boldsymbol{x}^\top \boldsymbol{y})^{2p} = (\mathbb{E}[|\boldsymbol{z}^\top \boldsymbol{x} \overline{\boldsymbol{z}}^\top \boldsymbol{y}|^2])^p - (\boldsymbol{x}^\top \boldsymbol{y})^{2p}. \qquad (62)\end{aligned}$$

Henceforth we focus on $\mathbb{E}[|\boldsymbol{z}^\top \boldsymbol{x} \overline{\boldsymbol{z}}^\top \boldsymbol{y}|^2]$ in the last expression (62). Write $\boldsymbol{z} = (z_1, \ldots, z_d)^\top$, $\boldsymbol{x} = (x_1, \ldots, x_d)^\top$, and $\boldsymbol{y} = (y_1, \ldots, y_d)^\top$. Since $\mathbb{E}[\boldsymbol{z}\overline{\boldsymbol{z}}^\top] = \boldsymbol{I}_d$, we have $\mathbb{E}[z_i \overline{z_j}] = 1$ if $i = j$ and $\mathbb{E}[z_i \overline{z_j}] = 0$ if $i \neq j$. Recall also that $z_1, \ldots, z_d \in \mathbb{C}$ are i.i.d, and $\mathbb{E}[z_i] = 0$ for $i = 1, \ldots, d$. Then

$$\begin{aligned}\mathbb{E}[|\boldsymbol{z}^\top \boldsymbol{x} \overline{\boldsymbol{z}}^\top \boldsymbol{y}|^2] &= \mathbb{E}\left[(\sum_{i=1}^{d} z_i x_i)(\sum_{j=1}^{d} \overline{z_j} y_j)(\sum_{k=1}^{d} \overline{z_k} x_k)(\sum_{l=1}^{d} z_l y_l) \right] \\ &= \sum_{i=1}^{d}\sum_{j=1}^{d}\sum_{k=1}^{d}\sum_{l=1}^{d} \mathbb{E}[z_i \overline{z_j}\,\overline{z_k} z_l] x_i y_j x_k y_l. \qquad (63)\end{aligned}$$

The expected value $\mathbb{E}\big[z_i \overline{z_j}\,\overline{z_k} z_l\big]$ is different from 0, only if:

(a) $i = j = k = l$, for which there are d terms and $\mathbb{E}\big[z_i \overline{z_j}\,\overline{z_k} z_l\big] x_i y_j x_k y_l = \mathbb{E}[|z_i|^4] x_i^2 y_i^2$.

(b) $i = j \neq k = l$, for which there are $d(d-1)$ terms and $\mathbb{E}\big[z_i \overline{z_j}\,\overline{z_k} z_l\big] x_i y_j x_k y_l = \mathbb{E}[|z_i|^2]\mathbb{E}[|z_k|^2] x_i x_k y_i y_k$.

(c) $i = k \neq j = l$, for which there are $d(d-1)$ terms and $\mathbb{E}\big[z_i \overline{z_j}\,\overline{z_k} z_l\big] x_i y_j x_k y_l = \mathbb{E}[|z_i|^2]\mathbb{E}[|z_j|^2] x_i^2 y_j^2$.

(d) $i = l \neq j = k$, for which there are $d(d-1)$ terms and $\mathbb{E}\big[z_i \overline{z_j}\,\overline{z_k} z_l\big] x_i y_j x_k y_l = \mathbb{E}[z_i^2]\mathbb{E}[\overline{z_j}^2] x_i x_j y_i y_j$.

Therefore,

$$
\begin{aligned}
(63) &= \underbrace{\sum_{i=1}^{d} \mathbb{E}[|z_i|^4] x_i^2 y_i^2}_{\text{case (a)}} + \underbrace{\sum_{i=1}^{d} \sum_{\substack{j=1 \\ j \neq i}}^{d} \mathbb{E}[|z_i|^2]\mathbb{E}[|z_j|^2] x_i^2 y_j^2}_{\text{case (c)}} + \underbrace{\sum_{i=1}^{d} \sum_{\substack{j=1 \\ j \neq i}}^{d} \mathbb{E}[|z_i|^2]\mathbb{E}[|z_j|^2] x_i x_j y_i y_j}_{\text{case (b)}} \\
&\quad + \underbrace{\sum_{i=1}^{d} \sum_{\substack{j=1 \\ j \neq i}}^{d} \mathbb{E}[z_i^2]\mathbb{E}[\overline{z_j}^2] x_i x_j y_i y_j}_{\text{case (d)}} \\
&= \sum_{i=1}^{d} \mathbb{E}[|z_i|^4] x_i^2 y_i^2 + \sum_{i=1}^{d} \sum_{\substack{j=1 \\ j \neq i}}^{d} x_i^2 y_j^2 + \sum_{i=1}^{d} \sum_{\substack{j=1 \\ j \neq i}}^{d} x_i x_j y_i y_j + \sum_{i=1}^{d} \sum_{\substack{j=1 \\ j \neq i}}^{d} \mathbb{E}[z_i^2]\mathbb{E}[\overline{z_j}^2] x_i x_j y_i y_j \\
&= \sum_{i=1}^{d} \mathbb{E}[|z_i|^4] x_i^2 y_i^2 + \left[\|\boldsymbol{x}\|^2 \|\boldsymbol{y}\|^2 - \sum_{i=1}^{d} x_i^2 y_i^2 \right] + \left[(\boldsymbol{x}^\top \boldsymbol{y})^2 - \sum_{i=1}^{d} x_i^2 y_i^2 \right] \\
&\quad + \sum_{i=1}^{d} \sum_{\substack{j=1 \\ j \neq i}}^{d} \mathbb{E}[z_i^2]\mathbb{E}[\overline{z_j}^2] x_i x_j y_i y_j
\end{aligned}
$$

The proof of Eq. (61) completes by using this expression of $\mathbb{E}[(\boldsymbol{z}^\top \boldsymbol{x} \overline{\boldsymbol{z}^\top \boldsymbol{y}})^2]$ in Eq. (62). Eq. (16) follows from Eq. (61) and $\mathbb{E}[z_k^2] = \mathbb{E}[\overline{z_k}^2] = 2q - 1$, which uses Eq. (15).

A.2 Proof of Theorem 3

We make use of Bernstein's inequality (e.g., Vershynin, 2018, Theorem 2.8.4): For independent random variables $X_1, \dots, X_D \in \mathbb{R}$ such that $\mathbb{E}[X_i] = 0$ and $|X_i| \leq R$ almost surely for a constant $R > 0$, we have for any $t > 0$:

$$
\Pr\left[\left| \sum_{i=1}^{D} X_i \right| \geq t \right] \leq 2 \exp\left(\frac{-t^2/2}{\sum_{i=1}^{D} \mathbb{V}[X_i] + Rt/3} \right) \tag{64}
$$

We define $X_i := \Phi_{\mathcal{C}}(\boldsymbol{x})_i \overline{\Phi_{\mathcal{C}}(\boldsymbol{y})_i} - (\boldsymbol{x}^\top \boldsymbol{y})^p / D \in \mathbb{R}$, where $\Phi_{\mathcal{C}}(\boldsymbol{x}) \in \mathbb{C}^D$ is defined in Eq. (10): $\Phi_{\mathcal{C}}(\boldsymbol{x}) = \frac{1}{\sqrt{D}} \left[(\prod_{i=1}^{p} \boldsymbol{z}_{i,1}^\top \boldsymbol{x}), \dots, (\prod_{i=1}^{p} \boldsymbol{z}_{i,D}^\top \boldsymbol{x}) \right]^\top$. Then we have $\mathbb{E}[X_i] = 0$. Moreover,

$$
\begin{aligned}
|X_i| &\leq |\Phi_{\mathcal{C}}(\boldsymbol{x})_i \overline{\Phi_{\mathcal{C}}(\boldsymbol{y})_i}| + |(\boldsymbol{x}^\top \boldsymbol{y})^p / D| = \frac{1}{D} \left(\prod_{j=1}^{p} |\boldsymbol{z}_j^\top \boldsymbol{x}| |\overline{\boldsymbol{z}_j^\top \boldsymbol{y}}| + |(\boldsymbol{x}^\top \boldsymbol{y})^p| \right) \\
&\leq \frac{1}{D} (\|\boldsymbol{x}\|_1^p \|\boldsymbol{y}\|_1^p + \|\boldsymbol{x}\|_2^p \|\boldsymbol{y}\|_2^p) \leq \frac{2}{D} \|\boldsymbol{x}\|_1^p \|\boldsymbol{y}\|_1^p =: R
\end{aligned}
$$

where the first inequality is the triangle inequality. The second inequality uses Hölder's inequality (and that the absolute value of each element of $\boldsymbol{z}_j$ is 1) as well as the upper bound $\boldsymbol{x}^\top \boldsymbol{y} \leq \|\boldsymbol{x}\|_2 \|\boldsymbol{y}\|_2$. Furthermore, by assumption we have

$$\mathbb{V}[X_i] = \frac{\sigma^2 \left\|\boldsymbol{x}\right\|_2^{2p} \left\|\boldsymbol{y}\right\|_2^{2p}}{D^2} \leq \frac{\sigma^2 \left\|\boldsymbol{x}\right\|_1^{2p} \left\|\boldsymbol{y}\right\|_1^{2p}}{D^2}$$

for some $\sigma^2 \geq 0$. Therefore, using Eq. (64) and setting $t := \left\|\boldsymbol{x}\right\|_1^p \left\|\boldsymbol{y}\right\|_1^p \epsilon$, we have

$$\Pr\left[\left|\sum_{i=1}^{D} X_i\right| \geq \epsilon \left\|\boldsymbol{x}\right\|_1^p \left\|\boldsymbol{y}\right\|_1^p\right] \leq 2\exp\left(\frac{-D\epsilon^2/2}{\frac{2}{3}\epsilon + \sigma^2}\right)$$

Setting $D \geq 2(\frac{2}{3\epsilon} + \frac{\sigma^2}{\epsilon^2})\log(\frac{2}{\delta})$ and taking the complementary probability gives the desired result.

Appendix B. Proofs for Section 4

B.1 Key Lemma

First, we state a key lemma that is needed for deriving the variance of real and complex TensorSRHT. This result is essentially given in Choromanski et al. (2017, Proof of Proposition 8.2). However, their proof contains a typo missing the negative sign, and they use a different definition of the Hadamard matrix from ours. Therefore, for completeness, we state the result formally and provide a proof.

Lemma 12 *Let $d = 2^m$ for some $m \in \mathbb{N}$ and $\boldsymbol{H}_d = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_d) \in \{1, -1\}^{d \times d}$ be the unnormalized Hadamard matrix defined in Eq. (24), where $\boldsymbol{h}_\ell = (h_{\ell,1}, \ldots, h_{\ell,d})^\top \in \{1, -1\}^d$ for $\ell \in \{1, \ldots, d\}$. Let $\pi : \{1, \ldots, d\} \to \{1, \ldots, d\}$ be a uniformly random permutation. Then for any $\ell, \ell' \in \{1, \ldots, d\}$ with $\ell \neq \ell'$ and $t, u \in \{1, \ldots, d\}$ with $t \neq u$, we have*

$$\mathbb{E}[h_{\pi(\ell),t} h_{\pi(\ell'),t} h_{\pi(\ell),u} h_{\pi(\ell'),u}] = -\frac{1}{d-1},$$

where the expectation is with respect to the random permutation π.

Proof We first derive a few key identities needed for our proof. For simplicity of notation, define

$$\alpha_\ell := h_{\ell,t} h_{\ell,u}, \quad \ell \in \{1, \ldots, d\}.$$

Since any two distinct rows (and any two distinct columns) of $\boldsymbol{H}_d$ are orthogonal, we have

$$\sum_{\ell=1}^{d} \alpha_\ell = \sum_{\ell=1}^{d} h_{\ell,t} h_{\ell,u} = 0.$$

Since $\alpha_\ell \in \{-1, 1\}$, this identity implies that exactly $d/2$ elements in $\{\alpha_1, \ldots, \alpha_d\}$ are 1, and the rest are -1. Note that for each $\ell \in \{1, \ldots, d\}$ the randomly permuted index $\pi(\ell)$ takes values in $\{1, \ldots, d\}$ with equal probabilities. Therefore, the probability of $\alpha_{\pi(\ell)}$ being 1 and that of $\alpha_{\pi(\ell)}$ being -1 are equal:

$$\Pr(\alpha_{\pi(\ell)} = 1) = \Pr(\alpha_{\pi(\ell)} = -1) = 0.5.$$

Note that $\pi^b(\ell) \neq \pi^b(\ell')$ since $\ell \neq \ell'$ and π is a (random) permutation. Therefore, we have the following conditional probabilities:

$$\Pr(\alpha_{\pi(\ell')} = a \mid \alpha_{\pi(\ell)} = b) = \begin{cases} \frac{d/2-1}{d-1} & \text{if } a = b = 1 \text{ or } a = b = -1 \\ \frac{d/2}{d-1} & \text{if } a = 1, b = -1 \text{ or } a = -1, b = -1 \end{cases}$$

Using the above identities, we now prove the assertion:

$$\begin{aligned} &\mathbb{E}[h_{\pi^b(\ell),t} h_{\pi^b(\ell'),t} h_{\pi^b(\ell),u} h_{\pi^b(\ell'),u}] = \mathbb{E}[\alpha_{\pi(\ell)}\alpha_{\pi(\ell')}] \\ &= \Pr(\alpha_{\pi(\ell)} = 1)\mathbb{E}[\alpha_{\pi(\ell)}\alpha_{\pi(\ell')} \mid \alpha_{\pi(\ell)} = 1] + \Pr(\alpha_{\pi(\ell)} = -1)\mathbb{E}[\alpha_{\pi(\ell)}\alpha_{\pi(\ell')} \mid \alpha_{\pi(\ell)} = -1] \\ &= \frac{1}{2}\mathbb{E}[\alpha_{\pi(\ell')} \mid \alpha_{\pi(\ell)} = 1] - \frac{1}{2}\mathbb{E}[\alpha_{\pi(\ell')} \mid \alpha_{\pi(\ell)} = -1] \\ &= \frac{1}{2}\left(\frac{d/2-1}{d-1} - \frac{d/2}{d-1}\right) - \frac{1}{2}\left(\frac{d/2}{d-1} - \frac{d/2-1}{d-1}\right) = -\frac{1}{d-1}. \end{aligned}$$

■

B.2 Proof of Theorem 10

We first clarify the notation we use. Recall that our feature map $\Phi(\boldsymbol{x}) \in \mathbb{C}^D$ is given by

$$\Phi(\boldsymbol{x}) = \frac{1}{\sqrt{D}}\left[(\prod_{i=1}^{p} \boldsymbol{s}_{i,1}^\top \boldsymbol{x}), \ldots, (\prod_{i=1}^{p} \boldsymbol{s}_{i,D}^\top \boldsymbol{x})\right]^\top \in \mathbb{C}^D.$$

The random vectors $\boldsymbol{s}_{i,\ell} \in \mathbb{C}^d$ are independently generated blockwise, and there are $B := \lceil D/d \rceil$ blocks in total (and note that $D = (B-1)d + \mathrm{mod}(D,d)$): For each $i = 1, \ldots, p$,

$$\begin{aligned} &\underbrace{(\boldsymbol{s}_{i,1}, \ldots, \boldsymbol{s}_{i,d})}_{\text{Block 1}}, \underbrace{(\boldsymbol{s}_{i,d+1}, \ldots, \boldsymbol{s}_{i,2d})}_{\text{Block 2}}, \ldots, \\ &\qquad \underbrace{(\boldsymbol{s}_{i,(B-2)d+1}, \ldots, \boldsymbol{s}_{i,(B-1)d})}_{\text{Block } B-1}, \underbrace{(\boldsymbol{s}_{i,(B-1)d+1}, \ldots, \boldsymbol{s}_{i,(B-1)d+\mathrm{mod}(D,d)})}_{\text{Block } B} \\ &=: \underbrace{(\boldsymbol{s}_{i,1}^1, \ldots, \boldsymbol{s}_{i,d}^1)}_{\text{Block 1}}, \underbrace{(\boldsymbol{s}_{i,1}^2, \ldots, \boldsymbol{s}_{i,d}^2)}_{\text{Block 2}}, \ldots, \underbrace{(\boldsymbol{s}_{i,1}^{B-1}, \ldots, \boldsymbol{s}_{i,d}^{B-1})}_{\text{Block } B-1}, \underbrace{(\boldsymbol{s}_{i,1}^{B}, \ldots, \boldsymbol{s}_{i,\mathrm{mod}(D,d)}^{B})}_{\text{Block } B}, \end{aligned}$$

where we introduced in the second line a new notation:

$$\boldsymbol{s}_{i,\ell}^b := \boldsymbol{s}_{i,(b-1)d+\ell} \quad (b = 1, \ldots, B,\ \ell = 1, \ldots, d.).$$

Here b serves as the indicator of the b-th block. Thus, using this notation,

$$\boldsymbol{s}_{i,\ell}^b = \boldsymbol{z}_i^b \circ \boldsymbol{h}_{\pi^b(\ell)} \in \mathbb{C}^d \quad (\ell = 1, \ldots, d),$$

where $\boldsymbol{z}_i^b = (z_{i,1}^b, \ldots, z_{i,d}^b)^\top \in \mathbb{C}^d$ is a random vector whose elements $z_{i,1}^b, \ldots, z_{i,d}^b$ are i.i.d., and $\pi^b : \{1, \ldots, d\} \to \{1, \ldots, d\}$ is a random permutation of the indices. Note that $\boldsymbol{z}_i^b$ and

π^b are generated independently for each $b \in \{1, \dots, B\}$. Therefore, the random vectors $\boldsymbol{s}_{i,\ell}^b$ and $\boldsymbol{s}_{i,\ell'}^{b'}$ are statistically independent if they are from different blocks, i.e., if $b \neq b'$.

For each $b = 1, \dots, B$, define $\boldsymbol{z}^b = (z_1^b, \dots, z_d^b)^\top \in \mathbb{C}^d$ as a random vector independently and identically distributed as $\boldsymbol{z}_1^b, \dots, \boldsymbol{z}_p^b$. Define

$$\boldsymbol{s}_\ell^b := \boldsymbol{z}^b \circ \boldsymbol{h}_{\pi(\ell)}^b = (z_1^b h_{\pi^b(\ell),1}, \dots, z_d^b h_{\pi^b(\ell),d})^\top =: (s_{\ell,1}^b, \dots, s_{\ell,d}^b)^\top \in \mathbb{C}^d. \tag{65}$$

Then $\boldsymbol{s}_\ell^b$ is independently and identically distributed as $\boldsymbol{s}_{1,\ell}^b, \dots, \boldsymbol{s}_{p,\ell}^b$. Moreover, given the permutation π^b fixed, $\boldsymbol{s}_\ell^b$ is identically distributed as $\boldsymbol{z}^b$. This is because 1) $z_1^b, \dots, z_d^b$ are i.i.d., 2) each z_t^b is symmetrically distributed ($t = 1, \dots, d$), and 3) $h_{\pi^b(\ell),1}, \dots, h_{\pi^b(\ell),d} \in \{1, -1\}$.

Now let us start proving the assertion. We first have

$$\mathbb{V}[\hat{k}(\boldsymbol{x}, \boldsymbol{y})] = \mathbb{E}[|\hat{k}(\boldsymbol{x}, \boldsymbol{y})|^2] - |\mathbb{E}[\hat{k}(\boldsymbol{x}, \boldsymbol{y})]|^2 = \mathbb{E}[|\hat{k}(\boldsymbol{x}, \boldsymbol{y})|^2] - (\boldsymbol{x}^\top \boldsymbol{y})^{2p},$$

where the second identity follows from the approximate kernel being unbiased for both real and complex TensorSRHT. Thus, from now on we study the term $\mathbb{E}[|\hat{k}(\boldsymbol{x}, \boldsymbol{y})|^2]$.

For simplicity of notation, define $I_b := \{1, \dots, d\}$ for $b = 1, \dots, B-1$ and $I_b := \{1, \dots, \mathrm{mod}(D, d)\}$ for $b = B$. Since the approximate kernel can be written as

$$\hat{k}(\boldsymbol{x}, \boldsymbol{y}) := \Phi(\boldsymbol{x})^\top \overline{\Phi(\boldsymbol{y})} = \frac{1}{D} \sum_{\ell=1}^{D} \prod_{i=1}^{p} \left(\boldsymbol{s}_{i,\ell}^\top \boldsymbol{x}\right) \overline{\left(\boldsymbol{s}_{i,\ell}^\top \boldsymbol{y}\right)} = \frac{1}{D} \sum_{b=1}^{B} \sum_{\ell \in I_b} \prod_{i=1}^{p} \left(\boldsymbol{s}_{i,\ell}^{b\top} \boldsymbol{x}\right) \overline{\left(\boldsymbol{s}_{i,\ell}^{b\top} \boldsymbol{y}\right)}$$

its second moment can be written as

$$\begin{aligned}
\mathbb{E}[|\hat{k}(\boldsymbol{x}, \boldsymbol{y})|^2] &= \frac{1}{D^2} \sum_{b,b'=1}^{B} \sum_{\ell \in I_b} \sum_{\ell \in I_{b'}} \mathbb{E}\left[\prod_{i=1}^{p} \left(\boldsymbol{s}_{i,\ell}^{b\top} \boldsymbol{x}\right) \overline{\left(\boldsymbol{s}_{i,\ell}^{b\top} \boldsymbol{y}\right)\left(\boldsymbol{s}_{i,\ell'}^{b'\top} \boldsymbol{x}\right)} \left(\boldsymbol{s}_{i,\ell'}^{b'\top} \boldsymbol{y}\right)\right] \\
&= \frac{1}{D^2} \sum_{b,b'=1}^{B} \sum_{\ell \in I_b} \sum_{\ell \in I_{b'}} \prod_{i=1}^{p} \mathbb{E}\left[\left(\boldsymbol{s}_{i,\ell}^{b\top} \boldsymbol{x}\right) \overline{\left(\boldsymbol{s}_{i,\ell}^{b\top} \boldsymbol{y}\right)\left(\boldsymbol{s}_{i,\ell'}^{b'\top} \boldsymbol{x}\right)} \left(\boldsymbol{s}_{i,\ell'}^{b'\top} \boldsymbol{y}\right)\right] \\
&= \frac{1}{D^2} \sum_{b,b'=1}^{B} \sum_{\ell \in I_b} \sum_{\ell \in I_{b'}} \left(\mathbb{E}\left[\left(\boldsymbol{s}_{\ell}^{b\top} \boldsymbol{x}\right) \overline{\left(\boldsymbol{s}_{\ell}^{b\top} \boldsymbol{y}\right)\left(\boldsymbol{s}_{\ell'}^{b'\top} \boldsymbol{x}\right)} \left(\boldsymbol{s}_{\ell'}^{b'\top} \boldsymbol{y}\right)\right]\right)^p.
\end{aligned} \tag{66}$$

Now we study individual terms in (66), categorizing the indices $b, b' \in \{1, \dots, B\}$ and $\ell, \ell' \in \{1, \dots, d\}$ of indices into the following 3 cases:

1. $\underline{b = b' \text{ and } \ell = \ell' \ (D \text{ terms})}$: As mentioned earlier, conditioned on the permutation π^b, $\boldsymbol{s}_\ell^b$ is identically distributed as $\boldsymbol{z}^b$ (see the paragraph following Eq. (65)). Thus,

$$\begin{aligned}
&\mathbb{E}\left[\left(\boldsymbol{s}_\ell^{b\top} \boldsymbol{x}\right)^2 \left(\boldsymbol{s}_\ell^{b\top} \boldsymbol{y}\right)^2\right] = \mathbb{E}_{\pi^b}\left[\mathbb{E}\left[\left(\boldsymbol{s}_\ell^{b\top} \boldsymbol{x}\right)^2 \left(\boldsymbol{s}_\ell^{b\top} \boldsymbol{y}\right)^2 \mid \pi^b\right]\right] \\
&= \mathbb{E}_{\pi^b}\left[\mathbb{E}\left[\left(\boldsymbol{z}^{b\top} \boldsymbol{x}\right)^2 \left(\boldsymbol{z}^{b\top} \boldsymbol{y}\right)^2\right]\right] = \mathbb{E}\left[\left(\boldsymbol{z}^{b\top} \boldsymbol{x}\right)^2 \left(\boldsymbol{z}^{b\top} \boldsymbol{y}\right)^2\right] = \mathbb{E}\left[\left(\boldsymbol{z}^{\top} \boldsymbol{x}\right)^2 \left(\boldsymbol{z}^{\top} \boldsymbol{y}\right)^2\right],
\end{aligned}$$

where $\mathbb{E}_{\pi^b}$ denotes the expectation with respect to π^b and $\boldsymbol{z} \in \mathbb{C}^d$ is a random vector identically distributed as $\boldsymbol{z}^1, \dots, \boldsymbol{z}^B$.

2. $b = b'$ and $\ell \neq \ell'$ ($c(D,d)$ terms, where $c(D,d)$ is defined in Eq. (29)): This case requires a detailed analysis, which we will do below.

3. $b \neq b'$ (The rest of terms $= D^2 - D - c(D,d)$ terms): Since $\boldsymbol{s}_\ell^b$ and $\boldsymbol{s}_{\ell'}^{b'}$ are independent in this case, we have

$$\mathbb{E}\left[\left(\boldsymbol{s}_\ell^{b\top}\boldsymbol{x}\right)\overline{\left(\boldsymbol{s}_\ell^{b\top}\boldsymbol{y}\right)\left(\boldsymbol{s}_{\ell'}^{b'\top}\boldsymbol{x}\right)}\left(\boldsymbol{s}_{\ell'}^{b'\top}\boldsymbol{y}\right)\right] = \mathbb{E}\left[\left(\boldsymbol{s}_\ell^{b\top}\boldsymbol{x}\right)\overline{\left(\boldsymbol{s}_\ell^{b\top}\boldsymbol{y}\right)}\right]\mathbb{E}\left[\overline{\left(\boldsymbol{s}_{\ell'}^{b'\top}\boldsymbol{x}\right)}\left(\boldsymbol{s}_{\ell'}^{b'\top}\boldsymbol{y}\right)\right]$$
$$= \mathbb{E}[\hat{k}(\boldsymbol{x},\boldsymbol{y})]\overline{\mathbb{E}[\hat{k}(\boldsymbol{x},\boldsymbol{y})]} = (\boldsymbol{x}^\top\boldsymbol{y})^2,$$

where the last equality follows from the approximate kernel being unbiased.

We now analyze the case 2:

$$\mathbb{E}\left[\left(\boldsymbol{s}_\ell^{b\top}\boldsymbol{x}\right)\overline{\left(\boldsymbol{s}_\ell^{b\top}\boldsymbol{y}\right)\left(\boldsymbol{s}_{\ell'}^{b\top}\boldsymbol{x}\right)}\left(\boldsymbol{s}_{\ell'}^{b\top}\boldsymbol{y}\right)\right] = \sum_{t,u,w,v=1}^{d}\mathbb{E}[s_{\ell,t}^b\overline{s_{\ell,u}^b s_{\ell',v}^b}s_{\ell',w}^b]x_t y_u x_v y_w$$

$$= \sum_{t,u,w,v=1}^{d}\mathbb{E}[z_t^b\overline{z_u^b z_v^b}z_w^b]\ \underbrace{\mathbb{E}[h_{\pi^b(\ell),t}h_{\pi^b(\ell),u}h_{\pi^b(\ell'),v}h_{\pi^b(\ell'),w}]}_{=:E}\ x_t y_u x_v y_w$$

Note that we have $\mathbb{E}[z_t^b\overline{z_u^b z_v^b}z_w^b] = 0$ unless:

(a) $t = u = v = w$: $\mathbb{E}[z_t^b\overline{z_u^b z_v^b}z_w^b] = \mathbb{E}[|z_t^b|^4] = 1$ and $E = \mathbb{E}[h^2_{\pi^b(\ell),t}h^2_{\pi^b(\ell'),t}] = 1$.

(b) $t = u \neq v = w$: $\mathbb{E}[z_t^b\overline{z_u^b z_v^b}z_w^b] = \mathbb{E}[|z_t^b|^2|z_v^b|^2] = 1$ and $E = \mathbb{E}[h^2_{\pi^b(\ell),t}h^2_{\pi^b(\ell'),v}] = 1$.

(c) $t = v \neq u = w$: $\mathbb{E}[z_t^b\overline{z_u^b z_v^b}z_w^b] = \mathbb{E}[|z_t^b|^2|z_u^b|^2] = 1$ and $E = \mathbb{E}[h_{\pi^b(\ell),t}h_{\pi^b(\ell),u}h_{\pi^b(\ell'),t}h_{\pi^b(\ell'),u}]$.

(d) $t = w \neq u = v$: $\mathbb{E}[z_t^b\overline{z_u^b z_v^b}z_w^b] = \mathbb{E}[(z_t^b)^2(\overline{z_u^b})^2] = (2q-1)^2$ and $E = \mathbb{E}[h_{\pi^b(\ell),t}h_{\pi^b(\ell),u}h_{\pi^b(\ell'),u}h_{\pi^b(\ell'),t}]$.

Therefore, we have

$$\mathbb{E}\left[\left(\boldsymbol{s}_\ell^{\top}\boldsymbol{x}\right)\left(\boldsymbol{s}_\ell^{\top}\boldsymbol{y}\right)\left(\boldsymbol{s}_{\ell'}^{\top}\boldsymbol{x}\right)\left(\boldsymbol{s}_{\ell'}^{\top}\boldsymbol{y}\right)\right]$$
$$= \sum_{t=1}^{d}x_t^2y_t^2 + \sum_{t\neq v}x_ty_tx_vy_v + \sum_{t\neq u}\mathbb{E}[h_{\pi^b(\ell),t}h_{\pi^b(\ell'),t}h_{\pi^b(\ell),u}h_{\pi^b(\ell'),u}]\left(x_t^2y_u^2 + (2q-1)^2x_ty_tx_uy_u\right)$$
$$= (\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{1}{d-1}\sum_{t\neq u}\left(x_t^2y_u^2 + (2q-1)^2x_ty_tx_uy_u\right) \quad (\because \text{Lemma 12})$$
$$= (\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{V_q^{(1)}}{d-1},$$

where $V_q^{(1)} := \sum_{t\neq u}\left(x_t^2y_u^2 + (2q-1)^2x_ty_tx_uy_u\right)$ is Eq. (17) with $p = 1$, which is the variance of the unstructured polynomial sketch (10) with a single feature.

Now, using these identities in Eq. (66), the variance of the approximate kernel can be expanded as

$$
\begin{aligned}
&\mathbb{V}[\hat{k}(\boldsymbol{x},\boldsymbol{y})] = \mathbb{E}[\hat{k}(\boldsymbol{x},\boldsymbol{y})^2] - (\boldsymbol{x}^\top \boldsymbol{y})^{2p} \\
=&\frac{1}{D}\left(\mathbb{E}\left[\left(\boldsymbol{z}^\top \boldsymbol{x}\right)^2\left(\boldsymbol{z}^\top \boldsymbol{y}\right)^2\right]\right)^p + \frac{c(D,d)}{D^2}\left((\boldsymbol{x}^\top \boldsymbol{y})^2 - \frac{V_q^{(1)}}{d-1}\right)^p \\
&+ \frac{D^2 - D - c(D,d)}{D^2}(\boldsymbol{x}^\top \boldsymbol{y})^{2p} - (\boldsymbol{x}^\top \boldsymbol{y})^{2p} \\
=&\frac{1}{D}\left[\left(\mathbb{E}\left[\left(\boldsymbol{z}^\top \boldsymbol{x}\right)^2\left(\boldsymbol{z}^\top \boldsymbol{y}\right)^2\right]\right)^p - (\boldsymbol{x}^\top \boldsymbol{y})^{2p}\right] - \frac{c(D,d)}{D^2}\left[(\boldsymbol{x}^\top \boldsymbol{y})^{2p} - \left((\boldsymbol{x}^\top \boldsymbol{y})^2 - \frac{V_q^{(1)}}{d-1}\right)^p\right] \\
=&\frac{1}{D}V_q^{(p)} - \frac{c(D,d)}{D^2}\left[(\boldsymbol{x}^\top \boldsymbol{y})^{2p} - \left((\boldsymbol{x}^\top \boldsymbol{y})^2 - \frac{V_q^{(1)}}{d-1}\right)^p\right]
\end{aligned}
$$

where $V_q^{(p)} \geq 0$ is Eq. (17) with the considered value of the polynomial degree p, which is the variance of the unstructured polynomial sketch (10) with a single feature. This completes the proof.

Appendix C. Convex Surrogate Functions for TensorSRHT Variances

To extend the applicability of the Incremental Algorithm in Algorithm 2 to TensorSRHT, we derive here convex surrogate functions for the variances of TensorSRHT, To this end, we first analyze the variances of TensorSRHT in Appendix C.1. We then derive convex surrogate functions in Appendix C.2.

C.1 Analyzing the Variances of TensorSRHT

We first derive another form of the variance of TensorSRHT given in Eq. (33) of Theorem 10, which we will use in a later analysis. Let $\Phi_n : \mathbb{R}^d \to \mathbb{C}^D$ be a complex TensorSRHT sketch of degree $n \in \mathbb{N}$ satisfying the assumptions in Theorem 10 with $0 \leq q \leq 1$. For $q = 1$ we recover the real TensorSRHT and for $q = 1/2$ the complex one.

As shown in Appendix B.2, the approximate kernel of the complex TensorSRHT can be written as

$$
\hat{k}(\boldsymbol{x},\boldsymbol{y}) := \Phi_n(\boldsymbol{x})^\top \overline{\Phi_n(\boldsymbol{y})} = \frac{1}{D}\sum_{b=1}^{B}\sum_{\ell\in I_b}\prod_{i=1}^{n}\left(\boldsymbol{s}_{i,\ell}^{b\top}\boldsymbol{x}\right)\overline{\left(\boldsymbol{s}_{i,\ell}^{b\top}\boldsymbol{y}\right)},
$$

where $B := \lceil D/d \rceil$, $I_b := \{1,\ldots,d\}$ for $b = 1,\ldots,B-1$ and $I_b := \{1,\ldots,\mathrm{mod}(D,d)\}$ for $b = B$, and $\boldsymbol{s}_{i,\ell}^b \in \mathbb{C}^d$ are the structured random weights defined in Eq. (65). We can then write the variance of the approximate kernel as

$$
\begin{aligned}
\mathbb{V}[\hat{k}(\boldsymbol{x},\boldsymbol{y})] =& \frac{1}{D^2}\sum_{b=1}^{B}\mathbb{V}\left[\sum_{\ell\in I_b}\prod_{i=1}^{n}\left(\boldsymbol{s}_{i,\ell}^{b\top}\boldsymbol{x}\right)\overline{\left(\boldsymbol{s}_{i,\ell}^{b\top}\boldsymbol{y}\right)}\right] \\
=& \frac{1}{D^2}\sum_{b=1}^{B}\sum_{\ell\in I_b}\underbrace{\mathbb{V}\left[\prod_{i=1}^{n}\left(\boldsymbol{s}_{i,\ell}^{b\top}\boldsymbol{x}\right)\overline{\left(\boldsymbol{s}_{i,\ell}^{b\top}\boldsymbol{y}\right)}\right]}_{=\,V_q^{(n)}} \\
&+ \frac{1}{D^2}\sum_{b=1}^{B}\sum_{\substack{\ell,\ell'\in I_b,\\ \ell\neq\ell'}}\underbrace{\mathrm{Cov}\left(\prod_{i=1}^{n}\left(\boldsymbol{s}_{i,\ell}^{b\top}\boldsymbol{x}\right)\overline{\left(\boldsymbol{s}_{i,\ell}^{b\top}\boldsymbol{y}\right)},\prod_{i=1}^{n}\left(\boldsymbol{s}_{i,\ell'}^{b\top}\boldsymbol{x}\right)\overline{\left(\boldsymbol{s}_{i,\ell'}^{b\top}\boldsymbol{y}\right)}\right)}_{=:\,\mathrm{Cov}_q^{(n)}} \\
=& \frac{V_q^{(n)}}{D} + \frac{c(D,d)}{D^2}\mathrm{Cov}_q^{(n)} = \text{Eq. (33)}, \qquad (67)
\end{aligned}
$$

where $c(D,d) = \lfloor D/d \rfloor d(d-1) + \mathrm{mod}(D,d)(\mathrm{mod}(D,d)-1)$ and the last line follows from that the values of $V_q^{(n)}$ and $\mathrm{Cov}_q^{(n)}$ do not depend on the choice of ℓ, ℓ' and b (which can be shown from the arguments in Appendix B.2). Here, $V_q^{(n)}$ is the variance of the unstructured Rademacher sketch with a single feature in Eq. (17) with $p = n$, and $\mathrm{Cov}_q^{(n)}$ is the covariance for distinct indices ℓ, ℓ' inside each block b. By comparing Eq. (33) and Eq. (67), the concrete form of $\mathrm{Cov}_q^{(n)}$ is given by

$$
\mathrm{Cov}_q^{(n)} = -\left[(\boldsymbol{x}^\top\boldsymbol{y})^{2n} - \left((\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{V_q^{(1)}}{d-1}\right)^n\right]
$$

Eq. (67) is a useful representation of the variance of TensorSRHT in Eq. (33) for studying its (non-)convexity with respect to D. The following result shows a range of values of D for which Eq. (33) is convex.

Theorem 13 *The variance of the TensorSRHT sketch in Eq.* (33) *is convex and monotonically decreasing with respect to $D \in \{1,\dots,d\}$ and with respect to $D \in \{kd \mid k \in \mathbb{N}\}$.*

Proof If $D \in \{1,\dots,d\}$, we have $c(D,d) = D(D-1)$ in Eq. (67). Therefore, Eq. (67) is equal to

$$
\frac{1}{D}V_q^{(n)} + \left(1 - \frac{1}{D}\right)\mathrm{Cov}_q^{(n)} = \frac{1}{D}\left(V_q^{(n)} - \mathrm{Cov}_q^{(n)}\right) + \mathrm{Cov}_q^{(n)}. \qquad (68)
$$

For two random variables X, Y it generally holds that $|\mathrm{Cov}(X,Y)| \leq \sqrt{\mathbb{V}[X]\mathbb{V}[Y]}$ by the Cauchy-Schwarz inequality. Hence, we have $|\mathrm{Cov}_q^{(n)}| \leq V_q^{(n)}$ and thus $V_q^{(n)} - \mathrm{Cov}_q^{(n)} \geq 0$. Therefore, Eq. (68) is proportional to $1/D$ with a non-negative coefficient, and thus it is convex and monotonically decreasing for $D \in \{1,\dots,d\}$.

Next, suppose $D = kd$ for some $k \in \mathbb{N}$, in which case we have $c(D,d) = kd(d-1)$ in Eq. (67). Therefore Eq. (67) is equal to

$$
\frac{1}{kd}\left(V_q^{(n)} + (d-1)\mathrm{Cov}_q^{(n)}\right) = \frac{1}{D}\left(V_q^{(n)} + (d-1)\mathrm{Cov}_q^{(n)}\right). \qquad (69)
$$

The term in the parenthesis is non-negative, because (67) is the variance of TensorSRHT and thus non-negative. Therefore, (67) is convex and monotonically decreasing with respect to $D \in \{kd \mid k \in \mathbb{N}\}$.

■

As we do next, Theorem 13 is useful for designing a convex surrogate function for Eq. (33), as it shows the range of D on which Eq. (33) is already convex and does not need to be modified.

C.2 Convex Surrogate Functions

Based on Eq. (33), we now propose a convex surrogate function for the variance of TensorSRHT in Eq. (33). We consider the following two cases separately: i) $\mathrm{Cov}_q^{(n)} \leq 0$ and ii) $\mathrm{Cov}_q^{(n)} > 0$. For each case, we propose a convex surrogate function.

i) Case $\mathrm{Cov}_q^{(n)} \leq 0$. We define a surrogate function of Eq. (33) by concatenating the two expressions of Eq. (67) for $D \in \{1, \dots, d\}$ and $D \in \{kd \mid k \in \mathbb{N}\}$ given in Eq. (68) and Eq. (69), respectively, and extend their ranges to the entire domain $D \in \mathbb{N}$:

$$V_{\mathrm{Surr.}}^{(n)}(D) := \begin{cases} \frac{1}{D}\left(V_q^{(n)} - \mathrm{Cov}_q^{(n)}\right) + \mathrm{Cov}_q^{(n)} & \text{if } D \leq d \\ \frac{1}{D}\left(V_q^{(n)} + (d-1)\mathrm{Cov}_q^{(n)}\right) & \text{if } D > d. \end{cases} \tag{70}$$

ii) Case $\mathrm{Cov}_q^{(n)} > 0$. We use the expression (69) to define a surrogate function on $D \in \mathbb{N}$:

$$V_{\mathrm{Surr.}}^{(n)}(D) := \frac{1}{D}\left(V_q^{(n)} + (d-1)\mathrm{Cov}_q^{(n)}\right) \tag{71}$$

The convexity of Eq. (71) immediately follows from $V_q^{(n)} + (d-1)\mathrm{Cov}_q^{(n)} \geq 0$, which holds as we show in the proof of Theorem 13. Note that $\mathrm{Cov}_q^{(n)} > 0$ can only occur when n is even, as shown in Corollary 11 of Section 4.

We defined the surrogate function in Eq. (70) by interpolating the variances of TensorSRHT in Eq. (33) for $D \in \{1, \dots, d\}$ and $D \in \{kd \mid k \in \mathbb{N}\}$ and extending the domain to $\mathbb{N}$. In fact, for $D \in \{1, \dots, d\}$ and $D \in \{kd \mid k \in \mathbb{N}\}$, Eq. (70) is equal to Eq. (33), as shown in the proof of Theorem 13. Fig. 12 illustrates the convex surrogate function in Eq. (70) and the variance of TensorSRHT in (33) when $\mathrm{Cov}_q^{(n)} \leq 0$ holds.

Note that, as mentioned later in Remark 15, the surrogate function in Eq. (70) may not be convex over $D \in \mathbb{N}$ if the condition $\mathrm{Cov}_q^{(n)} \leq 0$ does not hold. This is why we defined another convex surrogate function as in Eq. (71) for the case $\mathrm{Cov}_q^{(n)} > 0$.

The following theorem shows that the surrogate function in Eq. (70) is convex in the considered case of i) $\mathrm{Cov}_q^{(n)} \leq 0$.

Theorem 14 *If* $\mathrm{Cov}_q^{(n)} \leq 0$, *Eq.* (70) *is convex with respect to* $D \in \mathbb{N}$.

Proof As shown in Theorem 13, $V_{\mathrm{Surr.}}^{(n)}(D) = \frac{1}{D}(V_q^{(n)} - \mathrm{Cov}_q^{(n)}) + \mathrm{Cov}_q^{(n)}$ is convex over $D \in \{1, \dots, d\}$. Likewise, $V_{\mathrm{Surr.}}^{(n)}(D) = \frac{1}{D}(V_q^{(n)} + (d-1)\mathrm{Cov}_q^{(n)})$ is convex over $D \in [d, \infty) \cap \mathbb{N}$, since $V_q^{(n)} + (d-1)\mathrm{Cov}_q^{(n)} \geq 0$ holds as we show in the proof of Theorem 13.

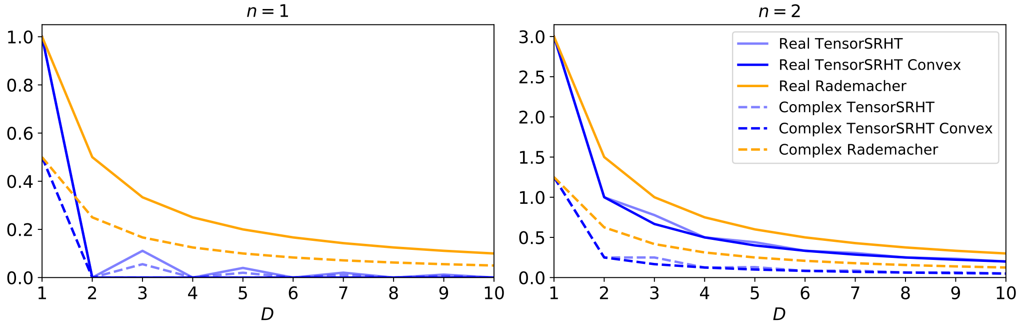

Figure 12: Convex surrogate functions in Eq. (70) and the variances of TensorSRHT in (33) as a function of the number of random features D, with polynomial degrees $n = 1, 2$ and input vectors $\boldsymbol{x} = \boldsymbol{y} = [\sqrt{1/2}, \sqrt{1/2}]^\top$ ($d = 2$). For comparison, we also plot the variances of the real Rademacher sketch in Eq. (18) and the complex Rademacher sketch in Eq. (19).

Therefore, the proof completes by showing that the concatenated function $V_{\text{Surr.}}^{(n)}(D)$ in Eq. (70) is also convex over $D \in \{d-1, d, d+1\}$, i.e.,

$$\frac{1}{2}\left(V_{\text{Surr.}}^{(n)}(d-1) + V_{\text{Surr.}}^{(n)}(d+1)\right) \geq V_{\text{Surr.}}^{(n)}(d). \tag{72}$$

By using the definition in Eq. (70), this inequality is equivalent to

$$\frac{1}{2}\left(\frac{1}{d-1}\left(V_q^{(n)} + (d-2)\text{Cov}_q^{(n)}\right) + \frac{1}{d+1}\left(V_q^{(n)} + (d-1)\text{Cov}_q^{(n)}\right)\right) \\ \geq \frac{1}{d}\left(V_q^{(n)} + (d-1)\text{Cov}_q^{(n)}\right). \tag{73}$$

Note that we have $V_q^{(n)} + (d-1)\text{Cov}_q^{(n)} \geq 0$, as mentioned earlier. If $V_q^{(n)} + (d-1)\text{Cov}_q^{(n)} = 0$ holds, then we have $V_{\text{Surr.}}^{(n)}(D) = 0$ for $D \geq d$ by the definition in Eq. (70), and thus Eq. (72) holds (which concludes the proofs). Therefore, we assume the inequality to be strict, i.e., $V_q^{(n)} + (d-1)\text{Cov}_q^{(n)} > 0$.

Dividing the both sides of Eq. (73) by $(V_q^{(n)} + (d-1)\text{Cov}_q^{(n)})$, we obtain

$$\frac{1}{2}\left(\frac{1}{d-1}\frac{V_q^{(n)} + (d-2)\text{Cov}_q^{(n)}}{V_q^{(n)} + (d-1)\text{Cov}_q^{(n)}} + \frac{1}{d+1}\right) \geq \frac{1}{d},$$

which after some rearrangement gives

$$\frac{V_q^{(n)} + (d-2)\text{Cov}_q^{(n)}}{V_q^{(n)} + (d-1)\text{Cov}_q^{(n)}} \geq 1 - \frac{2}{d^2 + d}. \tag{74}$$

This inequality holds because we have $(d-2)\text{Cov}_q^{(n)} \geq (d-1)\text{Cov}_q^{(n)}$, which follows from our assumption $\text{Cov}_q^{(n)} \leq 0$. Therefore Eq. (73) holds.

■

Remark 15 *Theorem 14 shows the convexity of the surrogate function in Eq.* (70)*, assuming* $\mathrm{Cov}_q^{(n)} \leq 0$*. If this condition does not hold, i.e., if* $\mathrm{Cov}_q^{(n)} > 0$*, then the surrogate function in Eq.* (70) *may not be convex. To see this, let* $d = 2, \boldsymbol{x} = (a, 0)^\top$ *with* $a > 0$*,* $\boldsymbol{y} = (0, b)^\top$ *with* $b > 0$*, and* n *be even; then we have* $V_q^{(n)} = \mathrm{Cov}_q^{(n)} = a^{2n} b^{2n} > 0$*, and the inequality in Eq.* (74) *in the proof of Theorem 14 does not hold, which implies that the surrogate function in Eq.* (70) *is not convex.*

As mentioned in Section 4, the variance of TensorSRHT in Eq. (33) becomes zero if $n = 1$ and $D \in \{kd \mid k \in \mathbb{N}\}$, i.e., $\mathbb{V}[\Phi_1(\boldsymbol{x})^\top \overline{\Phi_1(\boldsymbol{y})}] = 0$ holds. Therefore, because the convex surrogate functions in Eq. (70) and Eq. (71) are equal to the variance of TensorSRHT in Eq. (33) for $D \in \{kd \mid k \in \mathbb{N}\}$, these surrogate functions also become zero for $n = 1$ and $D \in \{kd \mid k \in \mathbb{N}\}$. Thus, the Incremental Algorithm (Algorithm 2), when used with the surrogate functions in Eq. (70) and Eq. (71), will not assign more than $D = d$ random features to the polynomial degree $n = 1$. Note that assigning $D = d$ random features is equivalent to appending the input vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ to the approximate kernel (44), which is called *H0/1 heuristic* in Kar and Karnick (2012). Therefore, the Incremental Algorithm with the surrogate functions in Eq. (70) and Eq. (71) automatically achieve the H0/1 heuristic.

Finally, we describe briefly how to use the convex surrogate functions in Eq. (70) and Eq. (71) in the Incremental Algorithm in Algorithm 2. To this end, we rewrite Eq. (54) using the surrogate functions as follows: (Here, we make the dependence of $V_q^{(n)}$ and $\mathrm{Cov}_q^{(n)}$ on the input vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ explicit and write them as $V_q^{(n)}(\boldsymbol{x}, \boldsymbol{y})$ and $\mathrm{Cov}_q^{(n)}(\boldsymbol{x}, \boldsymbol{y})$, respectively.)

$$\text{Eq. (54)} = \begin{cases} \frac{a_n^2}{D_n} \left(\sum_{i \neq j} V_q^{(n)}(\boldsymbol{x}_i, \boldsymbol{x}_j) + (d-1) \sum_{i \neq j} \mathrm{Cov}_q^{(n)}(\boldsymbol{x}_i, \boldsymbol{x}_j) \right) \\ \qquad \text{if } \sum_{i \neq j} \mathrm{Cov}_q^{(n)}(\boldsymbol{x}_i, \boldsymbol{x}_j) > 0 \text{ or } D_n > d, \\ \frac{a_n^2}{D_n} \left(\sum_{i \neq j} V_q^{(n)}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i \neq j} \mathrm{Cov}_q^{(n)}(\boldsymbol{x}_i, \boldsymbol{x}_j) \right) + a_n^2 \sum_{i \neq j} \mathrm{Cov}_q^{(n)}(\boldsymbol{x}_i, \boldsymbol{x}_j) \\ \qquad \text{otherwise.} \end{cases}$$

After precomputing the constants $\sum_{i \neq j} V_q^{(n)}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\sum_{i \neq j} \mathrm{Cov}_q^{(n)}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for each $n \in \{1, \ldots, p\}$, which can be done in $\mathcal{O}(m^2)$ time, one can directly use the above modification of Eq. (54) in the objective function in Eq. (53). In this way, we adapt the objective function in (53) to be convex, so that the Incremental Algorithm in Algorithm 2 is directly applicable.

Appendix D. Gaussian Processes with Complex Random Features

We describe here how to use complex random features in Gaussian process (GP) regression and classification. Since real random features are special cases of complex random features, all derivations for the complex case also hold for the real case as well.

For GP classification, we employ the framework of Milios et al. (2018), which formulates GP classification using GP regression and provides a solution in closed form. Therefore, closed form solutions are available for both GP regression and classification, and this enables us to compare different random feature approximations directly.[20]

[20] If we use a formulation of GP classification that requires an optimization procedure, comparisons of random feature approximations become more involved, as we need to perform convergence verification for the optimization procedure.

Notation and definitions. For a matrix $\boldsymbol{A} \in \mathbb{C}^{n \times m}$ with $n, m \in \mathbb{N}$, denote by $\boldsymbol{A}^H := \overline{\boldsymbol{A}}^\top \in \mathbb{C}^{m \times n}$ be its conjugate transpose. Note that if $\boldsymbol{A} \in \mathbb{R}^{n \times m}$, then $\boldsymbol{A}^H = \boldsymbol{A}^\top \in \mathbb{R}^{m \times n}$. For $n \in \mathbb{N}$, $\boldsymbol{I}_n \in \mathbb{R}^{n \times n}$ be the identity matrix.

For $\boldsymbol{\mu} \in \mathbb{C}^n$ and positive semi-definite[21] $\boldsymbol{\Sigma} \in \mathbb{C}^{n \times n}$ with $n \in \mathbb{N}$, we denote by $\mathcal{CN}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ the n-dimensional *proper* complex Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covaraince matrix $\boldsymbol{\Sigma}$, whose density function is given by (e.g., Neeser and Massey, 1993, Theorem 1)

$$\mathcal{CN}(\boldsymbol{v}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) := \frac{1}{\pi^n \sqrt{|\boldsymbol{\Sigma}|}} \exp\left(-(\boldsymbol{v} - \boldsymbol{\mu})^H \boldsymbol{\Sigma}^{-1} (\boldsymbol{v} - \boldsymbol{\mu})\right), \quad \boldsymbol{v} \in \mathbb{C}^n,$$

where $|\boldsymbol{\Sigma}|$ is the determinant of $\boldsymbol{\Sigma}$. If a random vector $\boldsymbol{f} \in \mathbb{C}^n$ follows $\mathcal{CN}(\boldsymbol{v}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$, we have $\mathbb{E}[\boldsymbol{f}] = \boldsymbol{\mu}$, $\mathbb{E}[(\boldsymbol{f} - \boldsymbol{\mu})(\boldsymbol{f} - \boldsymbol{\mu})^H] = \boldsymbol{\Sigma}$, and $\mathbb{E}[(\boldsymbol{f} - \boldsymbol{\mu})(\boldsymbol{f} - \boldsymbol{\mu})^\top] = 0$, where the last property is the definition of $\boldsymbol{f}$ being a proper complex random variable (Neeser and Massey, 1993, Definition 1).

D.1 Complex GP Regression

We first describe the approach of *complex GP regression* (Boloix-Tortosa et al., 2018), a Bayesian nonparametric approach to complex-valued regression.

Suppose that there are training data $(\boldsymbol{x}_i, y_i)_{i=1}^N \subset \mathbb{R}^d \times \mathbb{C}$ for a complex-valued regression problem with $N \in \mathbb{N}$, and let $\boldsymbol{X} := (\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)^\top \in \mathbb{R}^{N \times d}$ and $\boldsymbol{y} := (y_1, \dots, y_N)^\top \in \mathbb{C}^N$. We assume the following model for the training data:

$$y_i = f(x_i) + \varepsilon_i, \quad (i = 1, \dots, N), \tag{75}$$

where $f : \mathbb{R}^d \to \mathbb{C}$ is an unknown complex-valued function, and $\varepsilon_i \sim \mathcal{CN}(0, \sigma_i^2)$ is an independent complex Gaussian noise with variance $\sigma_i^2 > 0$. Let $\boldsymbol{\sigma}^2 := (\sigma_1^2, \dots, \sigma_N^2)^\top \in \mathbb{R}^N$.

The task of complex-valued function is to estimate the unknown complex-valued function f in Eq. (75) from the training data $(\boldsymbol{x}_i, y_i)_{i=1}^N \subset \mathbb{R}^d \times \mathbb{C}$. In complex GP regression, one defines a *complex GP prior distribution* for the unknown function f, and derives a *complex GP posterior distribution* of f, given the data $(\boldsymbol{x}_i, y_i)_{i=1}^N \subset \mathbb{R}^d \times \mathbb{C}$ and the likelihood function given by Eq. (75). For the prior, we focus on a *proper* complex GP (Boloix-Tortosa et al., 2018, Section II-C), which we describe below.

Proper complex Gaussian processes. A complex-valued function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ is called *positive definite kernel*, if 1) $k(\boldsymbol{x}, \boldsymbol{x}') = \overline{k(\boldsymbol{x}', \boldsymbol{x})}$ for all $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d$; and ii) for all $n \in \mathbb{N}$ and all $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$, the matrix $\boldsymbol{K} \in \mathbb{C}^{n \times n}$ with $\boldsymbol{K}_{i,j} = k(x_i, x_j)$ satisfies $\boldsymbol{v}^H \boldsymbol{K} \boldsymbol{v} \geq 0$.

Let $f : \mathbb{R}^d \to \mathbb{C}$ be a zero-mean complex-valued stochastic process, and $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ be a positive definite kernel. We call f a (zero-mean) *proper complex GP* with covariance kernel k, if for all $n \in \mathbb{N}$ and all $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^d$, the random vector $\boldsymbol{f} := (f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_n))^\top \in \mathbb{C}^n$ follows the proper complex Gaussian distribution $\mathcal{CN}(\boldsymbol{0}, \boldsymbol{K})$ with covariance matrix $\boldsymbol{K} \in \mathbb{C}^{n \times n}$ with $\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$. If f is a zero-mean proper complex GP with covariance kernel k, we write $f \sim \mathcal{CGP}(0, k)$.

We now describe the approach of complex GP regression. For the unknown f in Eq. (75), we define a proper complex GP prior with kernel k, assuming that

$$f \sim \mathcal{CGP}(0, k) \tag{76}$$

[21] A Hermitian matrix $\boldsymbol{\Sigma} \in \mathbb{C}^{n \times n}$ is called positive semi-definite, if for all $\boldsymbol{v} \in \mathbb{C}^n$, we have $\boldsymbol{v}^H \boldsymbol{\Sigma} \boldsymbol{v} \geq 0$.

Then the observation model (75) and the prior (76) induce a joint distribution of the unknown function f and the training observations $\boldsymbol{y} = (y_1, \ldots, y_N)^\top$. Conditioned on $\boldsymbol{y}$, we obtain the *posterior distribution* of f, which is also a proper complex GP (Boloix-Tortosa et al., 2018, Section II-C):

$$f \mid \boldsymbol{y} \sim \mathcal{CGP}(\mu_N, k_N), \tag{77}$$

where $\mu_N : \mathbb{R}^d \to \mathbb{C}$ is the *posterior mean function* and $k_N : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ is the *posterior covariance function* given by

$$\mu_N(\boldsymbol{x}) := \boldsymbol{k}(\boldsymbol{x})^H (\boldsymbol{K} + \operatorname{diag}(\boldsymbol{\sigma}^2))^{-1} \boldsymbol{y}, \quad \boldsymbol{x} \in \mathbb{R}^d \tag{78}$$

$$k_N(\boldsymbol{x}, \boldsymbol{x}') := k(\boldsymbol{x}, \boldsymbol{x}') - \boldsymbol{k}(\boldsymbol{x})^H (\boldsymbol{K} + \operatorname{diag}(\boldsymbol{\sigma}^2))^{-1} \boldsymbol{k}(\boldsymbol{x}), \quad \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d, \tag{79}$$

where $\boldsymbol{k}(\boldsymbol{x}) := (k(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_N))^\top \in \mathbb{C}^N$, $\boldsymbol{K} \in \mathbb{C}^{N \times N}$ with $\boldsymbol{K}_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and $\operatorname{diag}(\boldsymbol{\sigma}^2) \in \mathbb{R}^{d \times d}$ is the diagonal matrix with diagonal elements $\boldsymbol{\sigma}^2 = (\sigma_1^2, \ldots, \sigma_N^2)^\top$.

Notice that, if the kernel k is real-valued and so are the observations $\boldsymbol{y}$, Eq. (78) and Eq. (79) reduce to the posterior mean and covariance functions of standard real-valued GP regression (e.g., Rasmussen and Williams, 2006, Chapter 2). In this sense, complex GP regression with a proper GP prior is a natural complex extension of standard GP regression.

D.2 GP Regression with Complex Features

We next describe how to use complex features in GP regression. Let $\Phi : \mathbb{R}^d \to \mathbb{C}^D$ be a complex-valued (random) feature map,[22] and let $\hat{k}(\boldsymbol{x}, \boldsymbol{x}') := \Phi(\boldsymbol{x})^\top \overline{\Phi(\boldsymbol{x}')}$ be the approximate kernel. Define

$$\Phi(\boldsymbol{X}) := (\Phi(\boldsymbol{x}_1), \ldots, \Phi(\boldsymbol{x}_N))^\top \in \mathbb{C}^{N \times D}, \quad \hat{\boldsymbol{K}} := \Phi(\boldsymbol{X}) \Phi(\boldsymbol{X})^H \in \mathbb{C}^{N \times N}, \tag{80}$$

where $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N \in \mathbb{R}^D$ are training inputs. Note that $\hat{\boldsymbol{K}}_{i,j} = \Phi(\boldsymbol{x}_i)^\top \overline{\Phi(\boldsymbol{x}_j)} = \hat{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, i.e., $\hat{\boldsymbol{K}}$ is the kernel matrix with kernel $\hat{k}$.

The approximate kernel $\hat{k} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ is complex-valued, and thus induces a proper complex GP, $f \sim \mathcal{CGP}(0, \hat{k})$. Using this GP as a prior for the unknown function f in the observation model (75), and conditioning on the observations $\boldsymbol{y} = (y_1, \ldots, y_N)^\top$, we obtain the following approximate complex GP posterior:

$$f \mid \boldsymbol{y} \sim \mathcal{CGP}(\hat{\mu}_N, \hat{k}_N), \tag{81}$$

where $\hat{\mu}_N : \mathbb{R}^d \to \mathbb{C}$ is an approximate posterior mean function and $\hat{k}_N : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{C}$ is an approximate posterior covariance function, defined as

$$\hat{\mu}_N(\boldsymbol{x}) := \hat{\boldsymbol{k}}(\boldsymbol{x})^H (\hat{\boldsymbol{K}} + \operatorname{diag}(\boldsymbol{\sigma}^2))^{-1} \boldsymbol{y}, \quad \boldsymbol{x} \in \mathbb{R}^d \tag{82}$$

$$\hat{k}_N(\boldsymbol{x}, \boldsymbol{x}') := \hat{k}(\boldsymbol{x}, \boldsymbol{x}') - \hat{\boldsymbol{k}}(\boldsymbol{x})^H (\hat{\boldsymbol{K}} + \operatorname{diag}(\boldsymbol{\sigma}^2))^{-1} \hat{\boldsymbol{k}}(\boldsymbol{x}), \quad \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d, \tag{83}$$

where $\hat{\boldsymbol{k}}(\boldsymbol{x}) := (\hat{k}(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, \hat{k}(\boldsymbol{x}, \boldsymbol{x}_N))^\top \in \mathbb{C}^N$, and $\hat{\boldsymbol{K}} \in \mathbb{C}^{N \times N}$ with $\hat{\boldsymbol{K}}_{i,j} = \hat{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

Finally, we define a real-valued approximate GP posterior using the real parts of Eq. (82) and Eq. (83). That is, define $\hat{\mu}_{N,\mathbb{R}} : \mathbb{R}^d \to \mathbb{R}$ as the real part of the approximate posterior

[22] Again, this subsumes the case of real-valued feature maps.

mean function in Eq. (82) , and $\hat{k}_{N,\mathbb{R}}$ as the real part of the approximate covariance function in Eq. (83):

$$\hat{\mu}_{N,\mathbb{R}}(\boldsymbol{x}) := \mathcal{R}\left\{\hat{\mu}_N(\boldsymbol{x})\right\}, \quad \boldsymbol{x} \in \mathbb{R}^d, \tag{84}$$

$$\hat{k}_{N,\mathbb{R}}(\boldsymbol{x}, \boldsymbol{x}') := \mathcal{R}\left\{\hat{k}_N(\boldsymbol{x}, \boldsymbol{x}')\right\}, \quad \boldsymbol{x}, \boldsymbol{x}' \in \mathbb{R}^d. \tag{85}$$

Then, we define a real-valued GP with mean function $\hat{\mu}_{N,\mathbb{R}}$ and covariance function $\hat{k}_{N,\mathbb{R}}$:

$$f|\boldsymbol{y} \sim \mathcal{GP}(\hat{\mu}_{N,\mathbb{R}}, \hat{k}_{N,\mathbb{R}}).$$

We use this approximate GP for prediction tasks in our experiments.

Note that naive computations of Eq. (82) and Eq. (83) require $\mathcal{O}(N^3 + N^2 D)$ complexity, and thus do not leverage the computational advantage of random features. We will show next how to reformulate Eq. (82) and Eq. (83) to compute them in $\mathcal{O}(D^3 + ND^2)$, which is linear in the number of training data points N.

D.3 Computationally Efficient Implementation

We describe how to efficiently compute the approximate posterior mean and covariance functions in Eq. (82) and Eq. (83), respectively. To this end, recall the notation in Eq. (80). Let $\sigma^{-1} := (\sigma_1^{-1}, \dots, \sigma_N^{-1})^\top \in \mathbb{R}^N$ and $\sigma^{-2} := (\sigma_1^{-2}, \dots, \sigma_N^{-2})^\top \in \mathbb{R}^N$.

First we deal with Eq. (82). For a matrix $A \in \mathbb{C}^{N \times D}$, we have $(A^H A + \boldsymbol{I}_N)A^H = A^H(AA^H + \boldsymbol{I}_D)$, and thus $A^H(AA^H + \boldsymbol{I}_N)^{-1} = (A^H A + \boldsymbol{I}_D)^{-1}A^H$. By using this last identity with $A = \mathrm{diag}(\boldsymbol{\sigma}^{-1})\Phi(X) \in \mathbb{C}^{N \times D}$, we can rewrite Eq. (82) as

$$\begin{aligned}
\hat{\mu}_N(\boldsymbol{x}) &= \hat{\boldsymbol{k}}(\boldsymbol{x})^H(\hat{\boldsymbol{K}} + \mathrm{diag}(\boldsymbol{\sigma}^2))^{-1}\boldsymbol{y}, \\
&= \Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{X})^H \left(\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^H + \mathrm{diag}(\boldsymbol{\sigma}^2)\right)^{-1} \boldsymbol{y} \\
&= \Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{X})^H \,\mathrm{diag}(\boldsymbol{\sigma}^{-1}) \left(\mathrm{diag}(\boldsymbol{\sigma}^{-1})\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^H \,\mathrm{diag}(\boldsymbol{\sigma}^{-1}) + \boldsymbol{I}_N\right)^{-1} \mathrm{diag}(\boldsymbol{\sigma}^{-1})\boldsymbol{y} \\
&= \Phi(\boldsymbol{x})^\top \left(\Phi(\boldsymbol{X})^H \,\mathrm{diag}(\boldsymbol{\sigma}^{-2})\Phi(\boldsymbol{X}) + \boldsymbol{I}_D\right)^{-1} \Phi(\boldsymbol{X})^H \,\mathrm{diag}(\boldsymbol{\sigma}^{-2})\boldsymbol{y}.
\end{aligned} \tag{86}$$

Next we deal with Eq. (83). For matrices A, C, U, V of appropriate sizes with A invertible, the Woodbury matrix identity states that $A^{-1} - A^{-1}U(C^{-1} + VA^{-1}U)^{-1}VA^{-1} = (A + UCV)^{-1}$. By using the Woodbury identity with $A = \boldsymbol{I}_D$, $C = \mathrm{diag}(\boldsymbol{\sigma}^{-2})$, $U = \Phi(X)^H$ and $V = \Phi(X)$, we can rewrite Eq. (83) as

$$\begin{aligned}
\hat{k}_N(\boldsymbol{x}, \boldsymbol{x}') &= \hat{k}(\boldsymbol{x}, \boldsymbol{x}') - \hat{\boldsymbol{k}}(\boldsymbol{x})^H(\hat{\boldsymbol{K}} + \mathrm{diag}(\boldsymbol{\sigma}^2))^{-1}\hat{\boldsymbol{k}}(\boldsymbol{x}) \\
&= \Phi(\boldsymbol{x})^\top \overline{\Phi(\boldsymbol{x})} - \Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{X})^H \left(\Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^H + \mathrm{diag}(\boldsymbol{\sigma}^2)\right)^{-1} \Phi(\boldsymbol{X})\overline{\Phi(\boldsymbol{x})} \\
&= \Phi(\boldsymbol{x})^\top \left(\boldsymbol{I}_D - \Phi(\boldsymbol{X})^H \left(\mathrm{diag}(\boldsymbol{\sigma}^2) + \Phi(\boldsymbol{X})\Phi(\boldsymbol{X})^H\right)^{-1} \Phi(\boldsymbol{X})\right) \overline{\Phi(\boldsymbol{x})} \\
&= \Phi(\boldsymbol{x})^\top \left(\boldsymbol{I}_D + \Phi(\boldsymbol{X})^H \,\mathrm{diag}(\boldsymbol{\sigma}^{-2})\Phi(\boldsymbol{X})\right)^{-1} \overline{\Phi(\boldsymbol{x})}.
\end{aligned} \tag{87}$$

We now study the costs of computing Eq. (86) and Eq. (87). For both Eq. (86) and Eq. (87), the bottleneck is the computation of the inverse of the following matrix.

$$\boldsymbol{B} := \Phi(\boldsymbol{X})^H \,\mathrm{diag}(\boldsymbol{\sigma}^{-2})\Phi(\boldsymbol{X}) + \boldsymbol{I}_D \in \mathbb{C}^{D \times D}. \tag{88}$$

The time complexity of computing $\boldsymbol{B}$ is $\mathcal{O}(ND^2)$, and that of the inverse $\boldsymbol{B}^{-1}$ is $\mathcal{O}(D^3)$, the latter being the complexity of computing the Cholesky decomposition $\boldsymbol{B} = \boldsymbol{L}\boldsymbol{L}^H$ with $\boldsymbol{L} \in \mathbb{C}^{D\times D}$ being a lower triangular matrix. Thus, the overall cost of computing $\boldsymbol{B}^{-1}$ is $\mathcal{O}(ND^2 + D^3)$.

We next conduct a more detailed analysis of the costs of $\boldsymbol{B}$ and its Cholesky decomposition, and compare them with the computational costs for the corresponding matrix inversion using real-valued features (i.e., when $\Phi(X) \in \mathbb{R}^{N\times D}$). Below we use the shorthand $\tilde{\Phi}(\boldsymbol{X}) := \mathrm{diag}(\boldsymbol{\sigma}^{-1})\Phi(\boldsymbol{X})$ so that $\boldsymbol{B} = \tilde{\Phi}(\boldsymbol{X})^H\tilde{\Phi}(\boldsymbol{X}) + \boldsymbol{I}_D$. Then the real and imaginary parts of $\boldsymbol{B}$ can be written as

$$\begin{aligned}\mathcal{R}\{\boldsymbol{B}\} &= \mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})^H\tilde{\Phi}(\boldsymbol{X})\} + \boldsymbol{I}_D = \mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})\}^\top\mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})\} + \mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})\}^\top\mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})\} + \boldsymbol{I}_D\\ \mathcal{I}\{\boldsymbol{B}\} &= \mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})^H\tilde{\Phi}(\boldsymbol{X})\} = \mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})\}^\top\mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})\} - \mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})\}^\top\mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})\}.\end{aligned}$$

Since $(\mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})\}^\top\mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})\})^\top = \mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})\}^\top\mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})\}$, one can compute $\mathcal{I}\{\boldsymbol{B}\}$ by only computing $\mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})\}^\top\mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})\}$. Therefore, the computation of $\boldsymbol{B}$ requires the computations of the three real D-by-D matrices (i.e., $\mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})\}^\top\mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})\}$, $\mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})\}^\top\mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})\}$, and $\mathcal{R}\{\tilde{\Phi}(\boldsymbol{X})\}^\top\mathcal{I}\{\tilde{\Phi}(\boldsymbol{X})\}$). Thus, the total number of operations for computing $\boldsymbol{B}$ is $3\cdot(ND^2) + 2\cdot D^2$, where $3\cdot(ND^2)$ is operations for the matrix products and $2\cdot D^2$ for the addition and subtraction inside $\mathcal{R}\{\boldsymbol{B}\}$ and $\mathcal{I}\{\boldsymbol{B}\}$, respectively. Hence, assuming $N \gg D$, the computational cost for $\boldsymbol{B}$ is roughly 3 times more expensive than the corresponding cost when Φ is real-valued.

Computing the Cholesky decomposition of a D by D matrix requires roughly $\frac{1}{6}D^3$ subtractions and $\frac{1}{6}D^3$ multiplications (e.g., Trefethen and Bau, 1997, p. 175). Therefore, when Φ is real-valued (and thus $\boldsymbol{B}$ is real-valued), the Cholesky decomposition of $\boldsymbol{B}$ requires $\frac{1}{6}D^3 + \frac{1}{6}D^3 = \frac{1}{3}D^3$ FLOPS. On the other hand, when Φ is complex-valued, the Cholesky decomposition of $\boldsymbol{B}$ requires $\frac{4}{3}D^3$ FLOPS: one complex subtraction requires 2 real subtractions, and thus subtractions in total require $\frac{1}{6}D^3 \times 2 = \frac{1}{3}D^3$ FLOPS; one complex multiplication requires 4 real multiplications and 2 real subtractions, and thus multiplications in total require $\frac{1}{6}D^3 \times 6 = D^3$ FLOPS; thus $\frac{1}{3}D^3 + D^3 = \frac{4}{3}D^3$ FLOPS in total. Thus, the cost for computing the Cholesky decomposition of $\boldsymbol{B}$ when Φ is complex-valued is 4 times more expensive than the real-valued case.

The memory requirement for the complex case is 2 times as large as the real case, since the complex case requires storing both real and imaginary parts.

Note that, if one uses a $2D$-dimensional *real* feature map (i.e., $\Phi(\boldsymbol{X}) \in \mathbb{R}^{N\times 2D}$), then this requires 4 times as much memory, 4 times as many operations to compute the matrix $\boldsymbol{B}$, and 8 times as many operations for the Cholesky decomposition of $\boldsymbol{B}$ as those required for a D-dimensional real feature map. Therefore, using a $2D$-dimensional real feature map is computationally more expensive than using a D-dimensional complex feature map, since the latter only requires 2 times as much memory, 3 times as many operations for computing $\boldsymbol{B}$, and 4 times as many operations for computing the Cholesky decomposition of $\boldsymbol{B}$ as those required for a D-dimensional real feature map, as shown above. Note also that the performance improvement from using a D-dimensional complex feature map is typically larger than using a $2D$-dimensional real feature map; see the experiments in Section 6.

D.4 GP Classification as Closed-form Multi-output Regression

We now describe the GP classification approach of Milios et al. (2018), and how to use approximate posteriors for GP regegression in this approach.

We assume that there are $C \in \mathbb{N}$ classes and that output labels are expressed by one-hot encoding. Thus, for each class $c \in \{1, \ldots, C\}$ and each training input $\boldsymbol{x}_i \in \mathbb{R}^d$ with $i = 1, \ldots, N$, there exist an output $y_{c,i} \in \{0, 1\}$ such that $y_{c,i} = 1$ if $\boldsymbol{x}_i$ belongs to class c and $y_{c,i} = 0$ otherwise.

The approach of Milios et al. (2018). Let $\alpha > 0$ be a constant. For each class $c \in \{1, \ldots, C\}$, Milios et al. (2018) define transformed versions $\tilde{y}_{c,1}, \ldots, \tilde{y}_{c,N} \in \mathbb{R}$ of the training outputs $y_{c,1}, \ldots, y_{c,N}$ as

$$\tilde{y}_{c,i} := \log(y_{c,i} + \alpha) - \sigma_{c,i}^2/2, \quad \text{where} \quad \sigma_{c,i}^2 := \log((y_{c,i} + \alpha)^{-1} + 1), \quad i = 1, \ldots, N.$$

Milios et al. (2018) then define an observation model of $\tilde{y}_{c,1}, \ldots, \tilde{y}_{c,N}$ as

$$\tilde{y}_{c,i} = f_c(\boldsymbol{x}_i) + \varepsilon_{c,i}, \quad i = 1, \ldots, N, \tag{89}$$

where $f_c : \mathbb{R}^d \to \mathbb{R}$ is a latent function and $\varepsilon_{c,i} \sim \mathcal{N}(0, \sigma_{c,i}^2)$ is an independent Gaussian noise with variance $\sigma_{c,i}^2$. Milios et al. (2018) propose to model f_c for each $c \in \{1, \ldots, C\}$ independently as a GP:

$$f_c \sim \mathcal{GP}(0, k), \tag{90}$$

where $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ is a kernel. Eq. (89) and Eq. (90) define the joint distribution of the latent function f_c and the transformed labels $\tilde{y}_{c,1}, \ldots, \tilde{y}_{c,N}$. Thus, conditioning on $\tilde{y}_{c,1}, \ldots, \tilde{y}_{c,N}$, one obtains a GP posterior of f_c. In other words, one can obtain a GP posterior of f_c by performing GP regression for each class $c \in \{1, \ldots, C\}$ using $(\boldsymbol{x}_i, \tilde{y}_{c,i})_{i=1}^N$ as training data.

The constant α is a hyperparameter, which Milios et al. (2018) propose to choose by cross validation, using the Mean Negative Log Likelihood (MNLL) (e.g., Rasmussen and Williams, 2006, p. 23) as an evaluation criterion.

Using approximate GP posteriors. We now explain how to use approximate posteriors for GP regression in the above approach: For each class $c \in \{1, \ldots, C\}$, we perform approximate GP regression using $(\boldsymbol{x}_i, \tilde{y}_{c,i})_{i=1}^N$ as training data, to obtain an approximate GP posterior for the latent function f_c in Eq. (89). For instance, with our approach on approximate GP regression using complex random features in Appendix D.2, we obtain a GP posterior $f_c \sim \mathcal{GP}(\hat{\mu}_{N,\mathbb{R},c}, \hat{k}_{N,\mathbb{R},c})$ for each class $c \in \{1, \ldots, C\}$, where $\hat{\mu}_{N,\mathbb{R},c} : \mathbb{R}^d \to \mathbb{R}$ and $\hat{k}_{N,\mathbb{R},c} : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$ are the approximate GP posterior mean and covariance functions in Eq. (84) and Eq. (85), respectively, with $\boldsymbol{y} := (\tilde{y}_{c,1}, \ldots, \tilde{y}_{c,N})^\top$ and $\boldsymbol{\sigma}^2 := (\sigma_{c,1}^2, \ldots, \sigma_{c,N}^2)^\top$.

For a given test input $\boldsymbol{x} \in \mathbb{R}^d$, one can obtain its posterior predictive probabilities over the C classes in the following way. For each class $c \in \{1, \ldots, C\}$, we first generate a sample $z_c \in \mathbb{R}$ from the posterior distribution of the latent function value $f_c(\boldsymbol{x})$. We then apply the softmax transformation to $z_1, \ldots, z_C$ to obtain probabilities $p_1, \ldots, p_C \geq 0$ over the C class labels: $p_c := \exp(z_c) / \sum_{j=1}^C \exp(z_j)$. Milios et al. (2018) show that these probabilities $p_1, \ldots, p_C$ are approximately a sample from a Dirichlet distribution, yielding well-calibrated predictions.

D.5 Kullback-Leibler (KL) Divergence

In the experiments in Section 6, we use the Kullback-Leibler (KL) divergence between the exact and approximate GP posteriors, to evaluate the quality of each approximation approach. Let $\mu_{\text{exact}}(\boldsymbol{x})$ and $\sigma^2_{\text{exact}}(\boldsymbol{x})$ be the posterior mean and variance at $\boldsymbol{x} \in \mathbb{R}^d$ from the exact GP posterior, and let $\mu_{\text{appr}}(\boldsymbol{x})$ and $\sigma^2_{\text{appr}}(\boldsymbol{x})$ be those from an approximate GP posterior. Let $\boldsymbol{x}_{*,1}, \dots, \boldsymbol{x}_{*,m_*} \in \mathbb{R}^d$ be test input points. Define $\boldsymbol{\mu}_{\text{exact}} := (\mu_{\text{exact}}(\boldsymbol{x}_{*,1}), \dots, \mu_{\text{exact}}(\boldsymbol{x}_{*,m_*}))^\top$, $\boldsymbol{\sigma}^2_{\text{exact}} := (\sigma^2_{\text{exact}}(\boldsymbol{x}_{*,1}), \dots, \sigma^2_{\text{exact}}(\boldsymbol{x}_{*,m_*}))^\top$, $\boldsymbol{\mu}_{\text{appr}} := (\mu_{\text{appr}}(\boldsymbol{x}_{*,1}), \dots, \mu_{\text{appr}}(\boldsymbol{x}_{*,m_*}))^\top$, and $\boldsymbol{\sigma}^2_{\text{appr}} := (\sigma^2_{\text{appr}}(\boldsymbol{x}_{*,1}), \dots, \sigma^2_{\text{appr}}(\boldsymbol{x}_{*,m_*}))^\top$.

We then measure the KL divergence between two diagonal Gaussian distributions, $\mathcal{N}(\boldsymbol{\mu}_{\text{appr}}, \text{diag}(\boldsymbol{\sigma}^2_{\text{appr}}))$ and $\mathcal{N}(\boldsymbol{\mu}_{\text{exact}}, \text{diag}(\boldsymbol{\sigma}^2_{\text{exact}}))$:

$$\begin{aligned} &KL\left[\mathcal{N}(\boldsymbol{\mu}_{\text{appr}}, \text{diag}(\boldsymbol{\sigma}^2_{\text{appr}})) \mid\mid \mathcal{N}(\boldsymbol{\mu}_{\text{exact}}, \text{diag}(\boldsymbol{\sigma}^2_{\text{exact}}))\right] \\ &= \frac{1}{2}\sum_{i=1}^{m_*}\left(\frac{\sigma^2_{\text{exact}}(\boldsymbol{x}_{*,i})}{\sigma^2_{\text{appr}}(\boldsymbol{x}_{*,i})} + \log\frac{\sigma^2_{\text{exact}}(\boldsymbol{x}_{*,i})}{\sigma^2_{\text{appr}}(\boldsymbol{x}_{*,i})} - 1 + \frac{(\mu_{\text{exact}}(\boldsymbol{x}_{*,i}) - \mu_{\text{appr}}(\boldsymbol{x}_{*,i}))^2}{\sigma^2_{\text{appr}}(\boldsymbol{x}_{*,i})}\right), \end{aligned} \tag{91}$$

We consider these diagonal Gaussian distributions, since the focus of our experiments in Section 6 is the prediction performance at test input points $\boldsymbol{x}_{*,1}, \dots, \boldsymbol{x}_{*,m_*} \in \mathbb{R}^d$.

Appendix E. Additional Experiments

We present here additional experimental results, supplementing those in Section 6. Table 3 shows the effects of applying zero-centering to input vectors in the polynomial kernel approximation experiments. Fig. 13, Fig. 14 and Fig. 15 show the results of additional experiments on GP regression. Fig. 16 and Fig. 17 show the results of additional experiments on GP classification.

	MNLL				Rel. Frob. Error			
	Non-centred		Centred		Non-centred		Centred	
Dataset	SRF Gaus.	Opt. Macl. Rad.	SRF Gaus.	Opt. Macl. Rad.	SRF Gaus.	Opt. Macl. Rad.	SRF Gaus.	Opt. Macl. Rad.
Boston	3.410±0.37	3.447±0.38	3.449±0.62	**3.161**±0.28	**0.044**±0.02	0.212±0.15	0.356±0.05	0.421±0.06
Concrete	3.779±0.07	3.811±0.04	3.660±0.12	**3.542**±0.07	**0.019**±0.01	0.276±0.17	0.610±0.07	0.482±0.03
Energy	6.090±0.12	6.090±0.12	5.116±0.20	**5.012**±0.13	**0.003**±0.00	0.222±0.14	0.507±0.08	0.484±0.05
kin8nm	-0.203±0.07	-0.310±0.03	-0.203±0.07	**-0.323**±0.03	0.946±0.04	0.525±0.04	0.947±0.03	**0.521**±0.03
Naval	-6.069±0.03	-6.066±0.03	**-8.083**±0.04	-7.788±0.10	**0.040**±0.03	0.183±0.06	0.112±0.04	0.384±0.11
Powerplant	3.064±0.03	**3.061**±0.06	3.282±0.14	3.400±0.73	**0.001**±0.00	0.062±0.04	0.609±0.09	0.527±0.10
Protein	3.233±0.01	3.233±0.01	3.072±0.02	**3.060**±0.02	**0.000**±0.00	0.002±0.00	0.277±0.05	0.429±0.14
Yacht	4.317±0.45	4.478±0.45	**3.773**±0.21	3.844±0.28	**0.028**±0.01	0.276±0.11	0.512±0.04	0.484±0.03
Cod_rna	0.307±0.00	0.308±0.00	0.288±0.06	**0.151**±0.01	**0.022**±0.01	0.087±0.05	0.641±0.05	0.467±0.05
Covertype	0.821±0.01	-	0.650±0.01	**0.639**±0.01	**0.024**±0.01	-	0.361±0.01	0.300±0.01
Drive	1.446±0.02	1.453±0.03	0.677±0.02	**0.497**±0.01	**0.068**±0.02	0.135±0.05	0.348±0.01	0.312±0.02
FashionMNIST	**0.353**±0.00	0.364±0.00	0.364±0.00	0.361±0.00	**0.029**±0.00	0.062±0.01	0.099±0.00	0.104±0.01
Magic	0.453±0.01	0.452±0.01	0.381±0.02	**0.350**±0.01	**0.068**±0.01	0.147±0.05	0.430±0.03	0.418±0.04
Miniboo	0.253±0.01	-	0.239±0.01	**0.213**±0.01	**0.027**±0.01	-	0.214±0.01	0.229±0.02
MNIST	0.076±0.00	**0.074**±0.00	0.290±0.02	0.353±0.09	**0.073**±0.00	0.082±0.00	0.085±0.00	0.089±0.01
Mocap	0.360±0.01	0.334±0.01	0.357±0.02	**0.289**±0.01	**0.115**±0.01	0.187±0.04	0.414±0.01	0.290±0.02

Table 3: GP regression (top) and classification (bottom) for centred vs. non-centred data with $D = 5d$ features. Non-centred Miniboo and Covertype led to numerical issues for Maclaurin (no scores reported).

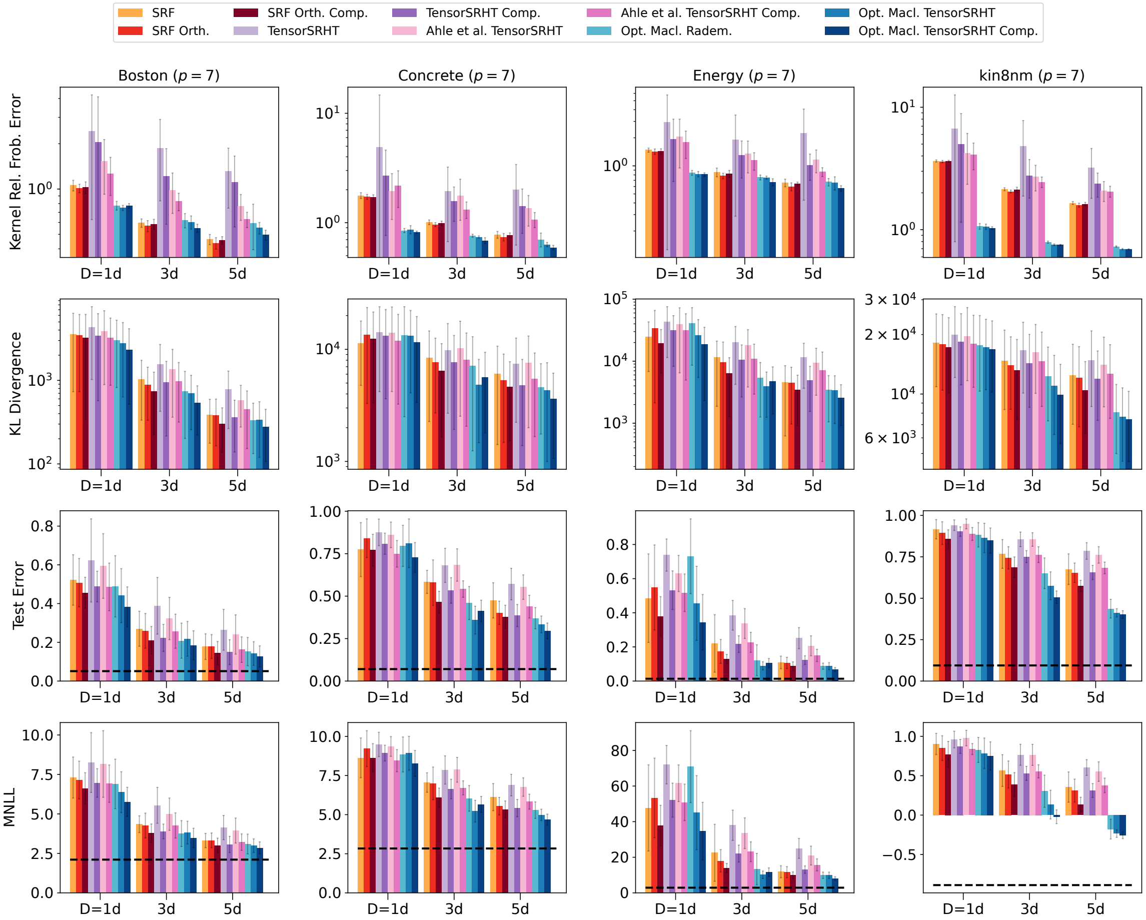

Figure 13: Additional results of the experiments in Section 6.4.1 on approximate GP regression with a $p = 7$ polynomial kernel. Lower values are better for all the metrics. For each dataset, we show the number of random features $D \in \{1d, 3d, 5d\}$ used in each method on the horizontal axis, with d being the input dimensionality of the dataset. The dashed black line shows test errors and MNLL values for the full target GP. In some cases performance is worse than the actual kernel because $D = 5d$ is still too small for some datasets. However, there is an indication that the test errors improve as we increase the number of features getting us closer to the true GP.

Appendix F. Additional Results for the Optimized Maclaurin Method

We present here additional results for Algorithm 3 in § 5.3. In Fig. 18 and Fig. 19 we analyze the output of the optimization phase involved in the Maclaurin approximation with Rademacher and TensorSRHT sketches, respectively. We focus on the approximation of the Gaussian kernel and determine p^* using $p_{\min} = 1$ and $p_{\max} = 20$. The algorithm is repeated for 20 different random seeds, and the resulting histogram of p^* is shown in the upper figures of Fig. 18 and Fig. 19; the bottom figures show the average and the standard deviation of the number of features assigned to each degree.

We observe that we need relatively few data samples (only 2000 out of 60000 training samples for FashionMNIST) to achieve a stable feature distribution. This supports our

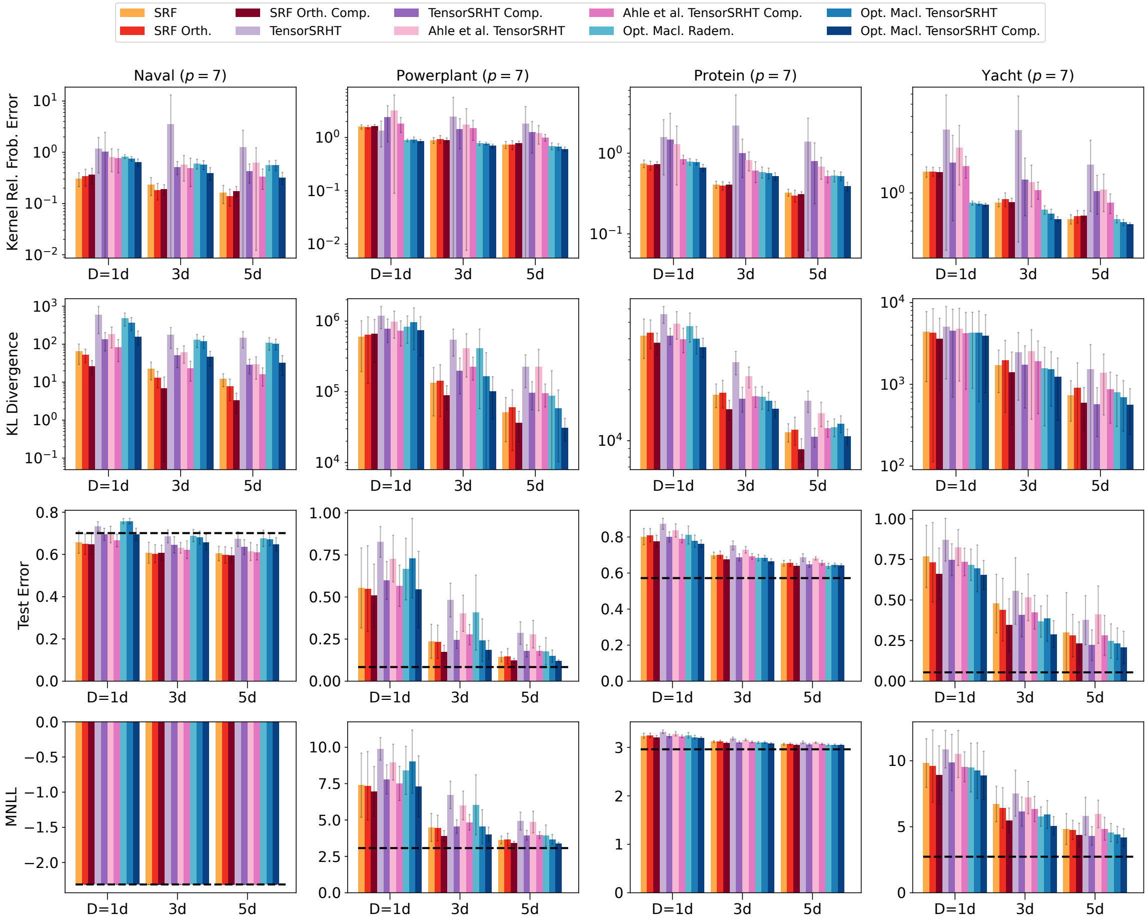

Figure 14: Additional results of the experiments in Section 6.4.1 on approximate GP regression with a $p = 7$ polynomial kernel. Lower values are better for all the metrics. For each dataset, we show the number of random features $D \in \{1d, 3d, 5d\}$ used in each method on the horizontal axis, with d being the input dimensionality of the dataset. The dashed black line shows test errors and MNLL values for the full target GP. Interestingly, in the Naval data set, the sketching approaches provide some form of regularization, yielding better generalization error than the GP with full kernel.

choice in the experiments in § 6, where we used considerably more (5000) data samples to estimate this. The value of p^* converges more slowly because sometimes very few (e.g., 1) random features are still allocated to high polynomial degrees. This does not harm the overall feature distribution too much since most features are already allocated to degrees 1-3 even for small sample sizes (e.g., 500 samples).

Interestingly, most features are allocated to low rather than high degrees, which is due to the variance distribution (see also Fig. 4). This supports our choice of $p_{\max} = 10$ in the experiments because most random features are allocated to small degrees. We set $p_{\min} = 2$ in the experiments to exclude purely linear approximations of the non-linear kernel. The low-degree allocation phenomenon was the same across datasets in Section 6 as long as the data is zero-centered. If it is not, this may change as shown in Wacker and Filippone (2021). Then high degrees may receive more features than lower ones.

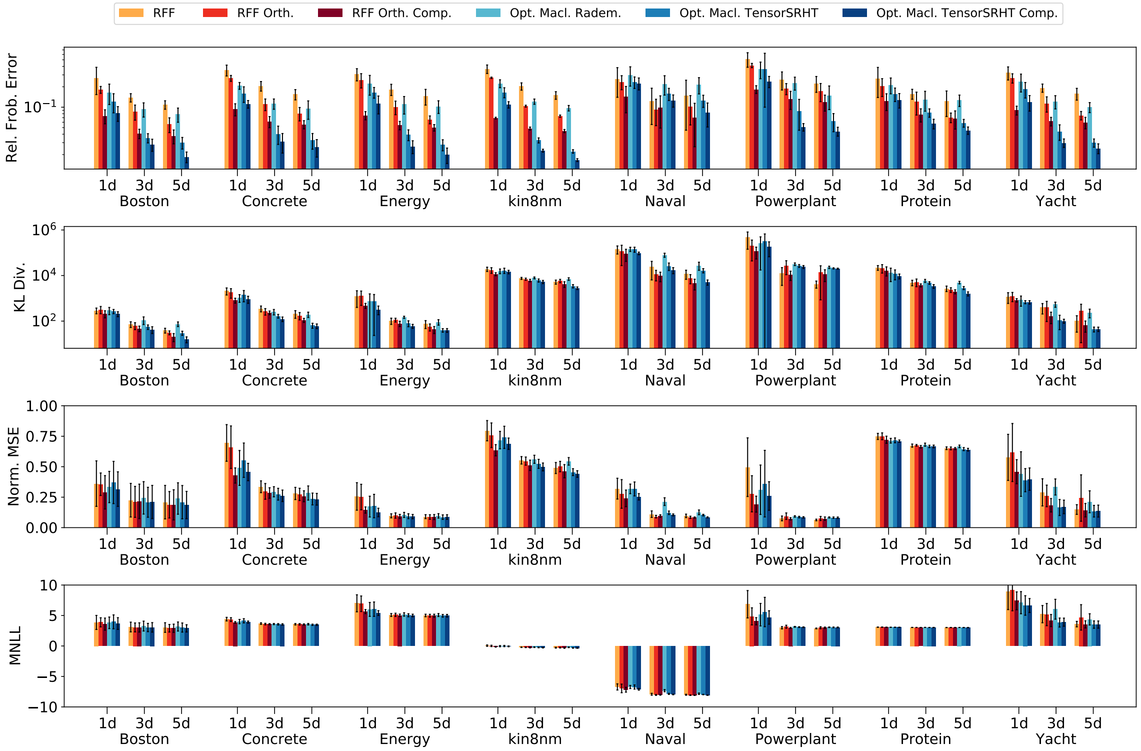

Figure 15: Additional results of the experiments in Section 6.4.2 on approximate GP regression with a Gaussian kernel. Lower values are better for all the metrics. For each dataset, we show the number of random features $D \in \{1d, 3d, 5d\}$ used in each method on the horizontal axis, with d being the input dimensionality of the dataset. We put the legend labels and the bars in the same order.

In the case of TensorSRHT, degree 1 is already perfectly approximated when $D_1 = d$ random features are used because the TensorSRHT variance always turns out to be zero for $p = 1$. The algorithm therefore starts investing random features into higher degrees once $D_1 = d$. So the next degree 2 is chosen to be the most dominant (for Rademacher it was degree 1). This explains the performance gain of Maclaurin TensorSRHT over Maclaurin Rademacher; we invest less random features into the first degree and use them to decrease the variances of other degrees.

Time Complexity for Different Values of $p_{\min}$ and $p_{\max}$. Here, we report a theoretical runtime analysis of the Maclaurin method. In Section 5.3, we have already reported an analysis of the incremental algorithm, giving a time complexity in $\mathcal{O}(pD_{\text{total}})$. In order to find the optimal p^*, we need to run the incremental algorithm $p_{\max} - p_{\min} + 1 \in \mathcal{O}(p_{\max})$ times. Additionally, the algorithm requires the *precomputed* variance estimates costing $\mathcal{O}(p_{\max} m^2)$, where m is the sample size. This gives a total time complexity of $\mathcal{O}(p_{\max}^2 D_{\text{total}} + p_{\max} m^2)$ for the feature allocation optimization.

Note that a simple Johnsson-Lindenstrauss projection costs $\mathcal{O}(ndD)$ for n data points, d input dimensions and D features. Note also that $p_{\max}^2$ is much smaller than nd though

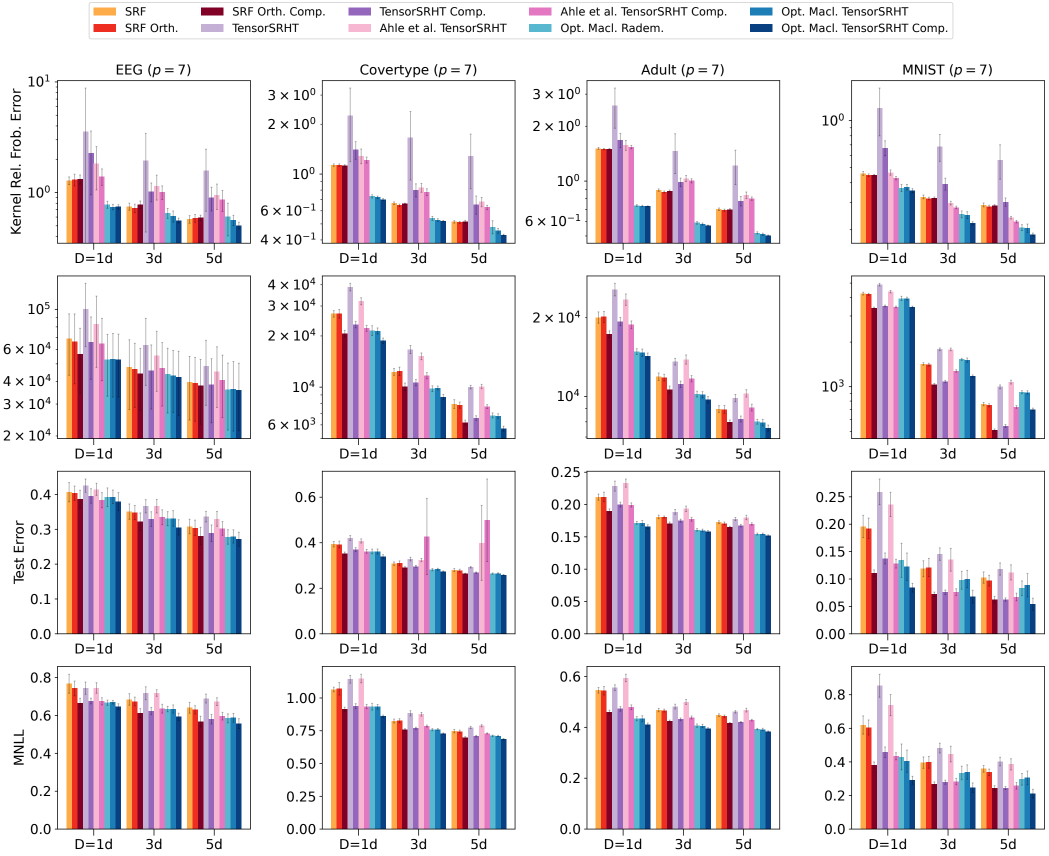

Figure 16: Additional results of the experiments in Section 6.4.1 on approximate GP classification with a $p = 7$ polynomial kernel. Lower values are better for all the metrics. For each dataset, we show the number of random features $D \in \{1d, 3d, 5d\}$ used in each method on the horizontal axis, with d being the input dimensionality of the dataset.

and $p_{\max} m^2$ is also small as long as $m \ll n$. The Maclaurin optimization has therefore a comparable time complexity to a Johnsson-Lindenstrauss projection for reasonable m.

References

M. Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *CoRR*, abs/1603.04467, 2016.

R. Agrawal, B. Trippe, J. Huggins, and T. Broderick. The kernel interaction trick: fast bayesian discovery of pairwise interactions in high dimensions. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 141–150. PMLR, 2019.

T. D. Ahle, M. Kapralov, J. B. T. Knudsen, R. Pagh, A. Velingker, D. P. Woodruff, and A. Zandieh. Oblivious sketching of high-degree polynomial kernels. In *Proceedings of*

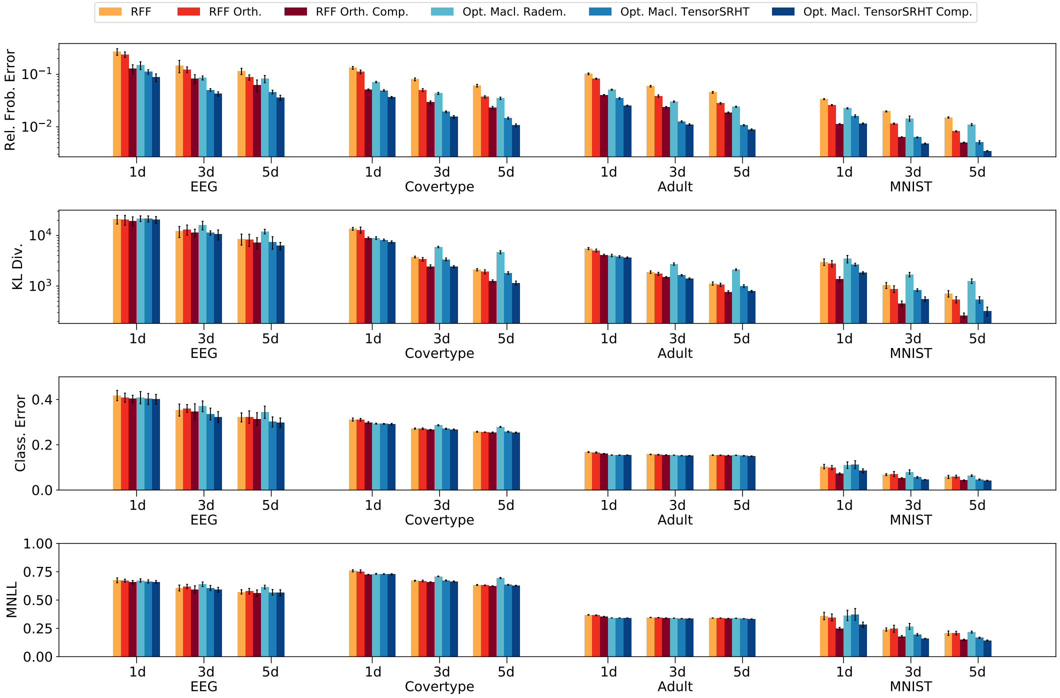

Figure 17: Additional results of the experiments in Section 6.4.2 on approximate GP classification with a Gaussian kernel. Lower values are better for all the metrics. For each dataset, we show the number of random features $D \in \{1d, 3d, 5d\}$ used in each method on the horizontal axis, with d being the input dimensionality of the dataset. We put the legend labels and the bars in the same order.

the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms, page 141–160. Society for Industrial and Applied Mathematics, 2020.

H. Aschard. A perspective on interaction effects in genetic association studies. *Genetic Epidemiology*, 40(8):678–688, 2016.

H. Avron, H. L. Nguyen, and D. P. Woodruff. Subspace embeddings for the polynomial kernel. In *Advances in Neural Information Processing Systems 27*, pages 2258–2266. Curran Associates, Inc., 2014.

M. Blondel, M. Ishihata, A. Fujino, and N. Ueda. Polynomial networks and factorization machines: New insights and efficient training algorithms. In *International Conference on Machine Learning*, pages 850–858. PMLR, 2016.

R. Boloix-Tortosa, J. J. Murillo-Fuentes, F. J. Payán-Somet, and F. Pérez-Cruz. Complex Gaussian processes for regression. *IEEE Transactions on Neural Networks and Learning Systems*, 29(11):5499–5511, 2018.

Y.-W. Chang, C.-J. Hsieh, K.-W. Chang, M. Ringgaard, and C.-J. Lin. Training and testing low-degree polynomial data mappings via linear SVM. *Journal of Machine Learning Research*, 11(4), 2010.

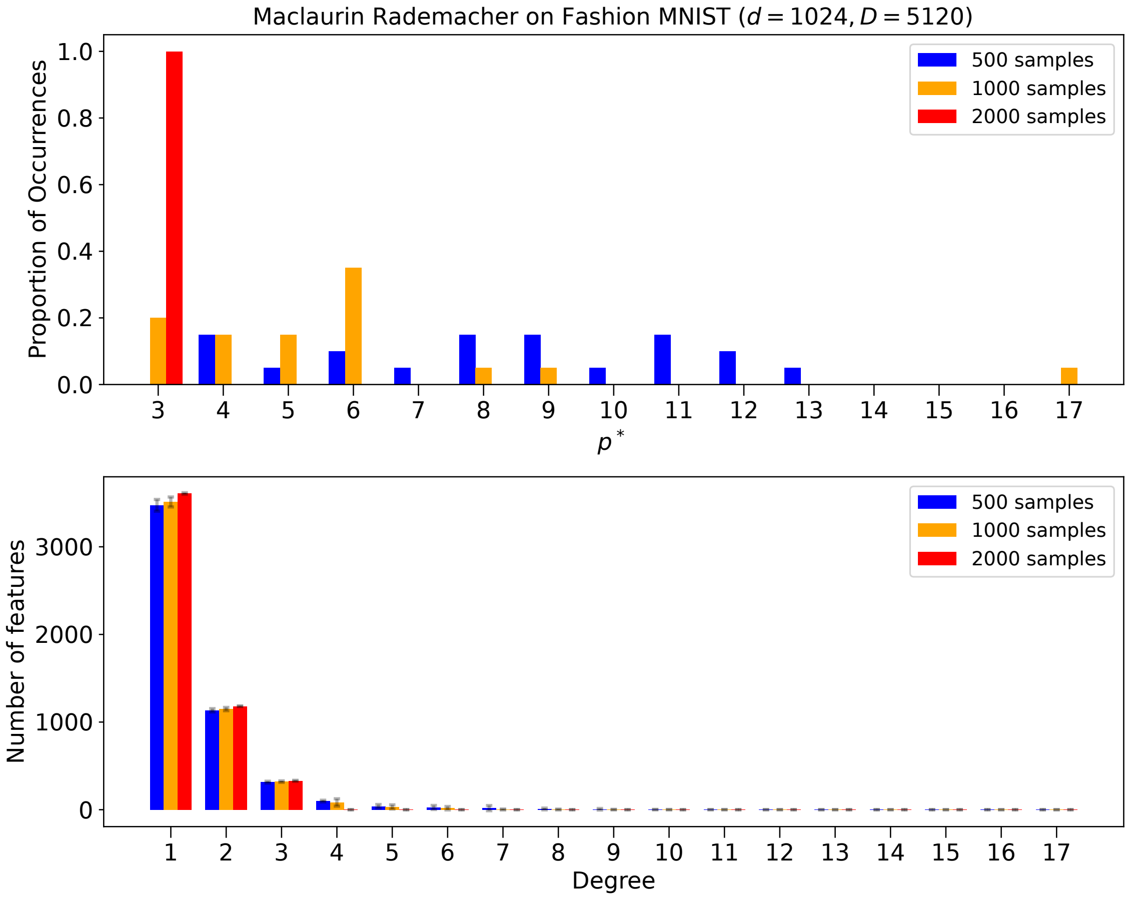

Figure 18: Comparing Maclaurin optimization outputs for different sample sizes 500, 1000 and 2000 and real Rademacher sketches over 20 runs of the algorithm. The optimal degree p^* approaches the value of 3 as the sample size increases (20/20 runs resulted in $p^* = 3$ for 2000 samples). The feature distribution across degrees allocates most random features to degree 1 followed by degree 2 and so on. This is already stable for small sample sizes (bar heights are mean values and error bars are standard deviations).

K. Choromanski, M. Rowland, and A. Weller. The unreasonable effectiveness of structured random orthogonal embeddings. In *Advances in Neural Information Processing Systems 31*, pages 218–227. Curran Associates Inc., 2017.

K. Choromanski, M. Rowland, T. Sarlos, V. Sindhwani, R. Turner, and A. Weller. The geometry of random features. In A. Storkey and F. Perez-Cruz, editors, *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, volume 84, pages 1–9. PMLR, 2018.

K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Q. Davis, A. Mohiuddin, L. Kaiser, D. B. Belanger, L. J. Colwell, and A. Weller. Rethinking attention with performers. In *Proceedings of the 9th International Conference on Learning Representations*, 2021.

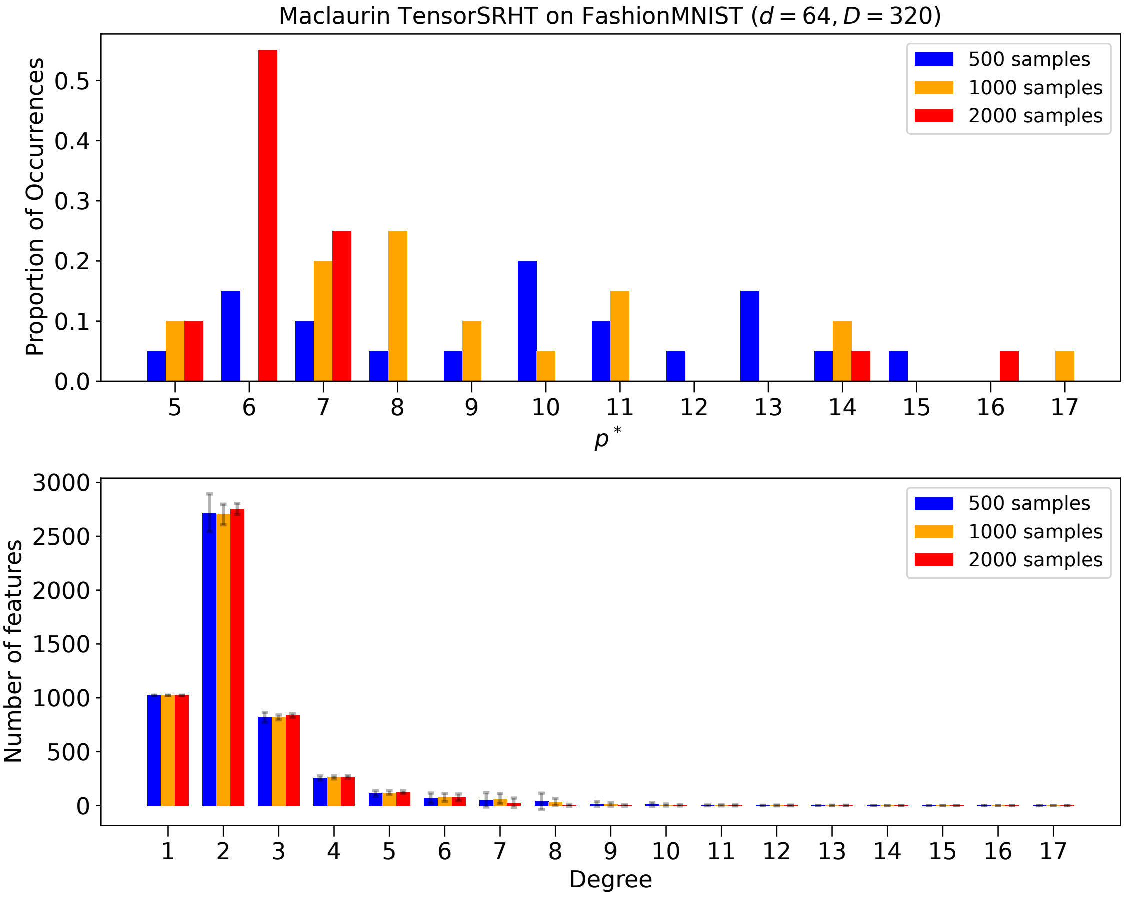

Figure 19: Comparing Maclaurin optimization outputs for different sample sizes 500, 1000 and 2000 and real TensorSRHT sketches. The optimal degree p^* approaches the value of 6 as the sample size increases. The feature distribution across degrees allocates most random features to degree 2 this time. This is already stable for small sample sizes.

A. Cotter, J. Keshet, and N. Srebro. Explicit approximations of the Gaussian kernel. *CoRR*, abs/1109.4603, 2011.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

B. J. Fino and V. R. Algazi. Unified matrix treatment of the fast Walsh-Hadamard transform. *IEEE Transactions on Computers*, 25(11):1142–1146, 1976.

C. A. Floudas and P. M. Pardalos, editors. *Encyclopedia of Optimization, Second Edition*. Springer, 2009.

A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach. Multimodal compact bilinear pooling for visual question answering and visual grounding. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 457–468. Association for Computational Linguistics, 2016.

Y. Gao, O. Beijbom, N. Zhang, and T. Darrell. Compact bilinear pooling. *Proceedings of the 2016 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 317–326, 2016.

D. Garreau, W. Jitkrittum, and M. Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017.

R. Hamid, Y. Xiao, A. Gittens, and D. DeCoste. Compact random feature maps. In *Proceedings of the 31th International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 19–27. PMLR, 2014.

C. R. Harris et al. Array programming with numpy. *Nature*, 585:357–362, 2020.

J. Hensman, N. Durrande, and A. Solin. Variational Fourier features for Gaussian processes. *Journal of Machine Learning Research*, 18(151):1–52, 2018.

P. Kar and H. Karnick. Random feature maps for dot product kernels. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics*, volume 22 of *JMLR Proceedings*, pages 583–591. JMLR, 2012.

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

T.-Y. Lin, A. RoyChowdhury, and S. Maji. Bilinear CNN models for fine-grained visual recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1449–1457, 2015.

F. Liu, X. Huang, Y. Chen, and J. A. K. Suykens. Random features for kernel approximation: A survey in algorithms, theory, and beyond. *CoRR*, abs/2004.11154, 2020.

D. Milios, R. Camoriano, P. Michiardi, L. Rosasco, and M. Filippone. Dirichlet-based Gaussian processes for large-scale calibrated classification. In *Advances in Neural Information Processing Systems 31*, pages 6008–6018. Curran Associates, Inc., 2018.

F. D. Neeser and J. L. Massey. Proper complex random processes with applications to information theory. *IEEE transactions on information theory*, 39(4):1293–1302, 1993.

A. Paszke et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8026–8037. Curran Associates, Inc., 2019.

J. Pennington, F. X. X. Yu, and S. Kumar. Spherical random features for polynomial kernels. In *Advances in Neural Information Processing Systems 28*, pages 1846–1854. Curran Associates, Inc., 2015.

N. Pham and R. Pagh. Fast and scalable polynomial kernels via explicit feature maps. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 239–247. Association for Computing Machinery, 2013.

A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems 20*, pages 1177–1184. Curran Associates Inc., 2007.

C. Rasmussen and C. Williams. *Gaussian Processes for Machine Learning.* MIT Press, 2006.

S. Rendle. Factorization machines. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, pages 995–1000, 2010.

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

A. Smola, Z. Óvári, and R. C. Williamson. Regularization with dot-product kernels. In *Advances in Neural Information Processing Systems 13*, pages 308–314. Curran Associates, Inc., 2000.

Z. Song, D. Woodruff, Z. Yu, and L. Zhang. Fast sketching of polynomial kernels of polynomial degree. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 9812–9823. PMLR, 2021.

D. J. Sutherland and J. Schneider. On the error of random fourier features. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 862–871. AUAI Press, 2015.

M. Titsias. Variational learning of inducing variables in sparse Gaussian processes. In *Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics*, volume 5 of *JMLR Proceedings*, pages 567–574. JMLR, 2009.

L. N. Trefethen and D. Bau. *Numerical Linear Algebra.* SIAM, 1997.

J. A. Tropp. Improved analysis of the subsampled randomized Hadamard transform. *Advances in Adaptive Data Analysis*, 3(1-2):115–126, 2011.

A. V. Uzilov, J. M. Keegan, and D. H. Mathews. Detection of non-coding RNAs on the basis of predicted secondary structure formation free energy change. *BMC Bioinformatics*, 7: 173, 2006.

A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017.

R. Vershynin. *High-Dimensional Probability: An Introduction with Applications in Data Science.* Cambridge University Press, 2018.

J. Wacker and M. Filippone. Local random feature approximations of the gaussian kernel. In *Proceedings of the 26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems*, Procedia Computer Science. Elsevier, 2021.

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

O. Weissbrod, D. Geiger, and S. Rosset. Multikernel linear mixed models for complex phenotype prediction. *Genome Research*, 26(7):969–979, 2016.

C. K. Williams and M. Seeger. Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. Curran Associates, Inc., 2000.

D. P. Woodruff. Sketching as a tool for numerical linear algebra. *Foundations and Trends in Theoretical Computer Science*, 10(1-2):1–157, 2014.

H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

H. Yamada and Y. Matsumoto. Statistical dependency analysis with support vector machines. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 195–206, 2003.

Z. Yang, M. Moczulski, M. Denil, N. D. Freitas, A. Smola, L. Song, and Z. Wang. Deep fried convnets. *Proceedings of the 2015 IEEE International Conference on Computer Vision*, pages 1476–1483, 2015.

F. X. Yu, A. T. Suresh, K. Choromanski, D. Holtmann-Rice, and S. Kumar. Orthogonal random features. In *Advances in Neural Information Processing Systems 30*, page 1983–1991. Curran Associates Inc., 2016.