Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 00 (2022) 000–000

Procedia Computer Science

www.elsevier.com/locate/procedia

26th International Conference on Knowledge-Based and Intelligent Information & Engineering Systems (KES 2022)

Local Random Feature Approximations of the Gaussian Kernel

Jonas Wacker*, Maurizio Filippone

Department of Data Science, EURECOM, 450 Route des Chappes, 06410 Biot, France

Abstract

A fundamental drawback of kernel-based statistical models is their limited scalability to large data sets, which requires resorting to approximations. In this work, we focus on the popular Gaussian kernel and on techniques to linearize kernel-based models by means of random feature approximations. In particular, we do so by studying a less explored random feature approximation based on Maclaurin expansions and polynomial sketches. We show that such approaches yield poor results when modelling high-frequency data, and we propose a novel localization scheme that improves kernel approximations and downstream performance significantly in this regime. We demonstrate these gains on a number of experiments involving the application of Gaussian process regression to synthetic and real-world data of different data sizes and dimensions.

Keywords: Random Features; Gaussian Kernel; Polynomial Kernel; Gaussian Processes

1. Introduction

Positive definite kernels are used to model nonlinear phenomena in a theoretically principled way. They have been extensively studied for kernel methods [17] as well as for Gaussian processes (GPs) [14], where they achieve competitive empirical performance [16]. For methods such as Gaussian process regression (GPR), we can obtain closed form predictions by solving linear systems, which is a substantial advantage over deep learning approaches that require iterative solvers and convergence verification. However, a naive application relies on algebraic operations on the kernel matrix (or Gram matrix) that consists of pairwise kernel evaluations of the training data. Constructing this matrix therefore requires $\mathcal{O}(N^2)$ computations and memory, where N is the number of training points, which obstructs the application of such models when the number of training points is large, e.g., $N > 10\,000$.

Hence, considerable effort has been dedicated to improving the scalability of kernel methods and GPs, in particular [22, 13, 19, 9]. All these methods turn the quadratic dependency on N into a linear or even sub-linear one when using mini-batching, which allows them to scale to millions of data points. A popular line of research uses so-called *random*

[*] Corresponding author.
E-mail address: jonas.wacker@eurecom.fr

feature (RF) approximations, which were originally introduced as *random Fourier features* for shift-invariant kernels [13] and were later extended to other classes of kernels such as dot product kernels [10].

Random feature approximations of the Gaussian kernel, which represents the focus of this work, are well-studied in the literature and are usually based on random Fourier features (c.f. [11] for a recent review). However, [3] have shown that the Gaussian kernel can also be formulated as a weighted sum of polynomial kernels allowing for a (non-random) Taylor series approximation of the exponential using explicit polynomial basis functions. [20] have further shown that using random feature approximations of these polynomial basis functions can make such approaches competitive with random Fourier features provided that the data is scaled appropriately.

In this work, we show that the approach in [20] fails for GPR when modelling high-frequency data for which the Gaussian kernel is parameterized by a short length scale. We propose a localized modification of the GPR predictor used in [20] that cures this pathology and show that our predictor is competitive and sometimes superior to random Fourier features for a given dimension of the feature map. We evaluate our novel predictor empirically on highly nonlinear synthetic and real-world data (typically modelled using short length scales), and show that it yields state-of-the-art performance regardless of the input dimension of the data. We made our code publicly available[1].

Our work is structured as follows. We cover GPR as well as its approximation through a Taylor series approximation with random features in Section 2. Our theoretical contributions are made in Section 3, where we identify and propose a cure for a pathology of such Taylor series approximations. The empirical evaluation is reported in Section 4.

2. Background on Gaussian process regression with Gaussian kernel approximations

2.1. Gaussian process regression (GPR)

Suppose a training data set $\mathcal{D} := \{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ with $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^d \times \mathbb{R}$ that we summarize in matrix notation as $\boldsymbol{X} := (\boldsymbol{x}_1, \dots, \boldsymbol{x}_N)^\top \in \mathbb{R}^{N \times d}$ and $\boldsymbol{y} := (y_1, \dots, y_N)^\top \in \mathbb{R}^N$. We assume that $\boldsymbol{y}$ has been generated from $\boldsymbol{X}$ by an unknown latent function $f : \mathbb{R}^d \to \mathbb{R}$ that has been corrupted by independent Gaussian noise, i.e., $y_i = f(\boldsymbol{x}_i) + \epsilon_i$ with $\epsilon_i \sim \mathcal{N}(0, \sigma^2_{\text{noise}})$ and $\sigma^2_{\text{noise}} > 0$.

In GPR [14, Chapter 2], the vector of function evaluations $\mathbf{f} := (f(\boldsymbol{x}_1), \dots, f(\boldsymbol{x}_N))^\top \in \mathbb{R}^N$ is assumed to have a joint Gaussian distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^N$ and covariance matrix $\boldsymbol{K}_{\mathbf{ff}} \in \mathbb{R}^{N \times N}$. We follow the standard approach and set $\boldsymbol{\mu}$ to zero here although more complex models exist [14, Chapter 2.7]. The entries of $\boldsymbol{K}_{\mathbf{ff}}$ correspond to the evaluations of a positive definite kernel function $k : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}$, i.e., $(\boldsymbol{K}_{\mathbf{ff}})_{i,j} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ that determines the covariance of a pair of function values $f(\boldsymbol{x}_i)$ and $f(\boldsymbol{x}_j)$.

The task of GPR is to predict the latent function value at a new test input $\boldsymbol{x}_* \in \mathbb{R}^d$ given the training set $\mathcal{D}$. The predictive distribution of $f(\boldsymbol{x}_*)|\mathcal{D} \in \mathbb{R}$ can be computed in closed form, and it is $\mathcal{N}(\mu_*, \sigma^2_*)$ with:

$$\mu_* := \boldsymbol{k}_{\mathbf{f}*}^\top (\boldsymbol{K}_{\mathbf{ff}} + \sigma^2_{\text{noise}} \boldsymbol{I})^{-1} \boldsymbol{y} \quad \text{and} \quad \sigma^2_* := k_{**} - \boldsymbol{k}_{\mathbf{f}*}^\top (\boldsymbol{K}_{\mathbf{ff}} + \sigma^2_{\text{noise}} \boldsymbol{I})^{-1} \boldsymbol{k}_{\mathbf{f}*}, \tag{1}$$

where $\boldsymbol{k}_{\mathbf{f}*} = (k(\boldsymbol{x}_1, \boldsymbol{x}_*), \dots, k(\boldsymbol{x}_N, \boldsymbol{x}_*))^\top \in \mathbb{R}^N$, $k_{**} = k(\boldsymbol{x}_*, \boldsymbol{x}_*)$ and $\boldsymbol{I} \in \mathbb{R}^{N \times N}$ is the identity matrix. GPR is an attractive modelling choice as it provides uncertainty estimates through the predictive variance σ^2_* next to the actual prediction μ_*. At the same time, the hyperparameters of the kernel function k can be obtained through a gradient-based optimization of the *log marginal likelihood* [14, Chapter 5.4] avoiding time-consuming cross-validation.

However, computing the GPR predictor can be expensive in practice. The computational bottleneck is to solve the linear systems in Eq. (1), which costs $O(N^3)$ time. Even storing the matrix $\boldsymbol{K}_{\mathbf{ff}}$ requires $O(N^2)$ memory and becomes infeasible in practice when N is large, typically greater than 10 000, and approximations become necessary.

Explicit feature space formulation. If there exists a finite-dimensional feature map $\Phi : \mathbb{R}^d \to \mathbb{R}^D$ such that $k(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{y})$, it can be shown [14, Chapter 2] that Eq. (1) can be reformulated as

$$\mu_* := \Phi(\boldsymbol{x}_*)^\top \boldsymbol{A}^{-1} \Phi(\boldsymbol{X})^\top \boldsymbol{y} / \sigma^2_{\text{noise}} \quad \text{and} \quad \sigma^2_* := \Phi(\boldsymbol{x}_*)^\top \boldsymbol{A}^{-1} \Phi(\boldsymbol{x}_*), \quad \text{with} \quad \boldsymbol{A} := \Phi(\boldsymbol{X})^\top \Phi(\boldsymbol{X}) / \sigma^2_{\text{noise}} + \boldsymbol{I}, \tag{2}$$

[1] https://github.com/joneswack/dp-rfs

where $\Phi(\boldsymbol{X}) = (\Phi(\boldsymbol{x}_1), \dots, \Phi(\boldsymbol{x}_N))^\top \in \mathbb{R}^{N \times D}$. The feature space representation (2) changes the computational cost to $O(ND^2)$ and thus improves the scaling of GPR drastically if $D \ll N$. Unfortunately, exact feature maps can be infinite dimensional and this holds in particular for the Gaussian kernel that we study in this work. However, there exist finite dimensional feature maps that yield an *approximate* Gaussian kernel and we discuss them next.

2.2. Truncated Maclaurin approximation of the Gaussian kernel

The Gaussian kernel for two inputs $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$ ($\boldsymbol{y}$ is different from the labels in Eq. (1) here) is defined as $k(\boldsymbol{x}, \boldsymbol{y}) = \sigma^2 \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|^2/(2l^2))$ with its parameters being the length scale $l > 0$ and kernel variance $\sigma^2 > 0$. We rewrite this kernel as a weighted sum of *polynomial kernels* $(\boldsymbol{x}^\top \boldsymbol{y}/l^2)^n$ for $n \in \mathbb{N}$ by using $\|\boldsymbol{x} - \boldsymbol{y}\|^2 = \|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - 2(\boldsymbol{x}^\top \boldsymbol{y})$:

$$k(\boldsymbol{x}, \boldsymbol{y}) = \sigma^2 \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2l^2}\right) \exp\left(-\frac{\|\boldsymbol{y}\|^2}{2l^2}\right) \exp\left(\frac{\boldsymbol{x}^\top \boldsymbol{y}}{l^2}\right) = \sigma^2 \exp\left(-\frac{\|\boldsymbol{x}\|^2}{2l^2}\right) \exp\left(-\frac{\|\boldsymbol{y}\|^2}{2l^2}\right) \sum_{n=0}^{\infty} \frac{1}{n!} \left(\frac{\boldsymbol{x}^\top \boldsymbol{y}}{l^2}\right)^n, \tag{3}$$

where the second equality of Eq. (3) follows from the Maclaurin series (Taylor series around zero) of the exponential function. In the following, we obtain a *finite-dimensional* feature map for an approximate Gaussian kernel through *explicit* feature maps for polynomial kernels.

Explicit feature map of the polynomial kernel. Let $\boldsymbol{a} \otimes \boldsymbol{b} = \mathrm{vec}(\boldsymbol{a}\boldsymbol{b}^\top) \in \mathbb{R}^{d^2}$ be the vectorized outer product of two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$. We further define $\boldsymbol{a}^{(n)} := \boldsymbol{a} \otimes \dots \otimes \boldsymbol{a} \in \mathbb{R}^{d^n}$ to be the result of applying this operation a total number of $(n-1)$ times to a vector with itself. To simplify the notation, we absorb the length scale in the input data, i.e., we define the inputs $\tilde{\boldsymbol{x}} := \boldsymbol{x}/l$ and $\tilde{\boldsymbol{y}} := \boldsymbol{y}/l$. Then the polynomial kernel in Eq. (3) can be written as $(\boldsymbol{x}^\top \boldsymbol{y}/l^2)^n = (\tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{y}})^n = (\tilde{\boldsymbol{x}}^{(n)})^\top (\tilde{\boldsymbol{y}}^{(n)})$ [17, Proposition 2.1], where $\tilde{\boldsymbol{x}}^{(n)}$ and $\tilde{\boldsymbol{y}}^{(n)}$ are its explicit feature maps. We can now use the explicit feature maps for polynomial kernels to obtain an explicit feature map for the Gaussian kernel.

Explicit feature map of the Gaussian kernel. If we truncate the inifinite Maclaurin series in Eq. (3) to a finite degree $p \in \mathbb{N}$, we obtain an approximate Gaussian kernel $k_p(\boldsymbol{x}, \boldsymbol{y}) = \Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{y})$ with an explicit feature map defined as:

$$\Phi(\boldsymbol{x}) := \sigma \exp\left(-\|\tilde{\boldsymbol{x}}\|^2/2\right) \left(1, (\tilde{\boldsymbol{x}}^{(1)}/\sqrt{1!})^\top, \dots, (\tilde{\boldsymbol{x}}^{(p)}/\sqrt{p!})^\top\right)^\top \in \mathbb{R}^D \quad \text{with} \quad D = 1 + d^1 + \dots + d^p. \tag{4}$$

The approximation error $|k_p(\boldsymbol{x}, \boldsymbol{y}) - k(\boldsymbol{x}, \boldsymbol{y})|$ depends on the truncation degree p of the Maclaurin series in Eq. (3). If $\tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{y}} \in \mathbb{R}$ is far away from zero, a large p is needed for $\sum_{n=0}^p (\tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{y}})^n/n!$ to be an accurate estimate of $\exp(\tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{y}})$. As the dimension D of $\Phi(\boldsymbol{x})$ scales as $O(d^p)$, it becomes infeasible to construct such feature maps in practice when d or p are large. Thus, this approach was considered less efficient than random Fourier features [13] with respect to D in [3].

In the following, we substitute the explicit feature maps $\tilde{\boldsymbol{x}}^{(1)}, \dots, \tilde{\boldsymbol{x}}^{(p)}$ in Eq. (4) with low-dimensional *random* feature maps to attain good kernel estimates with reasonable D that become competitive with random Fourier features.

2.3. Optimized random Maclaurin features for the Gaussian kernel

Instead of using the explicit feature maps $\{\tilde{\boldsymbol{x}}^{(n)}\}_{n=1}^p$ in Eq. (4), [20] suggest to use *random* features for polynomial kernels as soon as $d > 1$. Here we consider random features that have been used in [10, 8, 20, 21]. Unlike for the ones proposed in [12, 1], there are closed form variance formulas available for the former in the literature [20] that allow us to optimize the variances of our kernel approximation in the following.

Polynomial Sketches. We define a random feature map ϕ_n, from now on called a *polynomial sketch*, as:

$$\phi_n(\boldsymbol{x}) := (\boldsymbol{W}_1 \boldsymbol{x} \odot \dots \odot \boldsymbol{W}_n \boldsymbol{x})/\sqrt{D} \in \mathbb{R}^D, \quad (\odot \text{ denotes the element-wise product}) \tag{5}$$

where $\boldsymbol{W}_1, \dots, \boldsymbol{W}_n \in \mathbb{R}^{D \times d}$ are i.i.d. random matrices. For two inputs $\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}} \in \mathbb{R}^d$ we have $\mathbb{E}[\phi_n(\tilde{\boldsymbol{x}})^\top \phi_n(\tilde{\boldsymbol{y}})] = (\tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{y}})^n$, i.e., the random feature approximation $\phi_n(\tilde{\boldsymbol{x}})^\top \phi_n(\tilde{\boldsymbol{y}})$ is an unbiased estimate of the polynomial kernel, if the $\{\boldsymbol{W}_i\}_{i=1}^n$ are sampled from an appropriate distribution. The variance of the approximation decreases as D increases. Table (1) shows three valid example sampling procedures for $\{\boldsymbol{W}\}_{i=1}^n$ along with the resulting variances $\mathbb{V}[\phi_n(\boldsymbol{x})^\top \phi_n(\boldsymbol{y})]$ of the corresponding kernel estimate.

Table 1. Polynomial Sketches according to [20] along with the variances of the associated kernel estimate. Here, $D \in \mathbb{N}$ is the number of random features and $c(D, d) := \lfloor D/d \rfloor d(d-1) + (D \mod d)(D \mod d-1)$.

Polynomial Sketch	Sampling Procedure for $\{\boldsymbol{W}_i\}_{i=1}^n$	Variance $\mathbb{V}[\phi_n(\boldsymbol{x})^\top \phi_n(\boldsymbol{y})]$
Gaussian	Entries of $\boldsymbol{W}_i$ are sampled i.i.d. from $\mathcal{N}(0,1)$	$D^{-1}[(\|\boldsymbol{x}\|^2\,\|\boldsymbol{y}\|^2 + 2(\boldsymbol{x}^\top\boldsymbol{y})^2)^n - (\boldsymbol{x}^\top\boldsymbol{y})^{2n}]$
Rademacher	Entries of $\boldsymbol{W}_i$ are sampled i.i.d. from $\mathrm{Unif}(\{1,-1\})$	$D^{-1}[(\|\boldsymbol{x}\|^2\,\|\boldsymbol{y}\|^2 + 2((\boldsymbol{x}^\top\boldsymbol{y})^2 - \sum_{i=1}^d x_i^2 y_i^2))^n - (\boldsymbol{x}^\top\boldsymbol{y})^{2n}]$
TensorSRHT	$\boldsymbol{W}_i$ is implicitly defined through Algorithm (2)	Rademacher variance $-\frac{c(D,d)}{D^2}\cdot$ $\left[(\boldsymbol{x}^\top\boldsymbol{y})^{2n} - \left((\boldsymbol{x}^\top\boldsymbol{y})^2 - \frac{1}{d-1}\left(\|\boldsymbol{x}\|^2\|\boldsymbol{y}\|^2 + (\boldsymbol{x}^\top\boldsymbol{y})^2 - 2\sum_{k=1}^d x_k^2 y_k^2\right)\right)^n\right]$

As shown in [20], the variance of the Rademacher polynomial sketch (second row in Table (1)) is upper-bounded by the variance of the Gaussian polynomial sketch (first row). TensorSRHT (third row) is a structured polynomial sketch that imposes an orthogonality constraint on the rows of each $\{\boldsymbol{W}\}_{i=1}^n$ leading to even lower variances for odd degrees n. We summarize its construction in Algorithm (2) in Appendix A. It uses the Fast Walsh-Hadamard Transform [5] to project a data point in $O(nD \log d)$ instead of $O(nDd)$ time required for Gaussian and Rademacher sketches.

In this work, we only make use of Rademacher and TensorSRHT sketches as they yield the lowest variances. It is also possible to use complex-valued random matrices in Eq. (5) that yield additional variance reductions for positively valued data [21]. However, this condition does not hold for the method proposed in this work, which is why we consider only real-valued polynomial sketches here.

Randomized Gaussian kernel approximation. We now consider using the set of polynomial sketches $\{\phi_n(\tilde{\boldsymbol{x}})\}_{n=1}^p$ (5) instead of the explicit feature maps $\{\tilde{\boldsymbol{x}}^{(n)}\}_{n=1}^p$ in $\Phi(\boldsymbol{x})$ (4), which yields the approximate Gaussian kernel:

$$\hat{k}_p(\boldsymbol{x},\boldsymbol{y}) := \Phi(\boldsymbol{x})^\top\Phi(\boldsymbol{y}) = \sigma^2 \exp\left(-\|\tilde{\boldsymbol{x}}\|^2/2\right)\exp\left(-\|\tilde{\boldsymbol{y}}\|^2/2\right)\left(1 + \sum_{n=1}^p \frac{1}{n!}\phi_n(\tilde{\boldsymbol{x}})^\top\phi_n(\tilde{\boldsymbol{y}})\right) \text{ with } \mathbb{E}[\hat{k}_p(\boldsymbol{x},\boldsymbol{y})] = k_p(\boldsymbol{x},\boldsymbol{y}), \quad (6)$$

where the expectation is with respect to the random feature distribution. The dimension of the feature map $\Phi(\boldsymbol{x})$ is $D = 1 + D_1 + \cdots + D_p$, where $\{D_n\}_{n=1}^p$ are the number of random features allocated to the polynomial sketches $\{\Phi_n\}_{n=1}^p$.

While the $\{D_n\}_{n=1}^p$ were chosen randomly in the past [10], it was shown in [20] that their allocation under a given budget $D-1$ has a significant impact on the quality of the kernel approximation. The authors in [20] show that the feature allocation task can be formulated as a discrete resource allocation problem for which the so-called *Incremental Algorithm* [6, p. 384] can be applied. Due to space limitations, we refer the reader to [20, Chapter 5.3 and Algorithm 3] that describes the procedure of finding an optimal degree p^* and an optical allocation $(D_1,\ldots,D_{p^*})$ using a subsample of the training data. We will use this method from now on whenever the input dimension d of the data is at least two, otherwise no approximation is needed and explicit polynomial feature maps Eq. (4) can be used.

3. Localized random Maclaurin features for the Gaussian kernel

In this section, we develop our main theoretical and methodological contribution. We begin by uncovering a pathology of Maclaurin-based approximations of the Gaussian kernel that appears when $\|\tilde{\boldsymbol{x}}\|$ or $\|\tilde{\boldsymbol{y}}\|$ become large.

3.1. Pathology of the Maclaurin method

We derive the following Theorem in Appendix B that characterizes the pathology of Maclaurin-based approximations leading to poor GPR predictions for high-frequency data.

Theorem 3.1 (Vanishing Maclaurin approximation of Gaussian kernels)**.** *The magnitude of the finite Maclaurin approximation $k_p(\boldsymbol{x},\boldsymbol{y}) = \sigma^2\exp(-(\|\tilde{\boldsymbol{x}}\|^2+\|\tilde{\boldsymbol{y}}\|^2)/2)\sum_{n=0}^p 1/n!(\tilde{\boldsymbol{x}}^\top\tilde{\boldsymbol{y}})^n$ of the Gaussian kernel approaches zero as $\|\tilde{\boldsymbol{x}}\|^2+\|\tilde{\boldsymbol{y}}\|^2$ increases. The error $|k(\boldsymbol{x},\boldsymbol{y}) - k_p(\boldsymbol{x},\boldsymbol{y})|$ between the exact kernel k and its approximation k_p is the largest for parallel $\boldsymbol{x},\boldsymbol{y}$ and zero when they are orthogonal.*

We visualize the implications of Theorem (3.1) in Fig. (1), where we compare the approximation $k_p(\boldsymbol{x},\boldsymbol{y})$ with the exact Gaussian kernel $k(\boldsymbol{x},\boldsymbol{y})$ for $p=3$ over a range of values $\|\tilde{\boldsymbol{x}}\|, \|\tilde{\boldsymbol{y}}\|$ as well as the angle θ_{xy} between $\boldsymbol{x}$ and $\boldsymbol{y}$. One

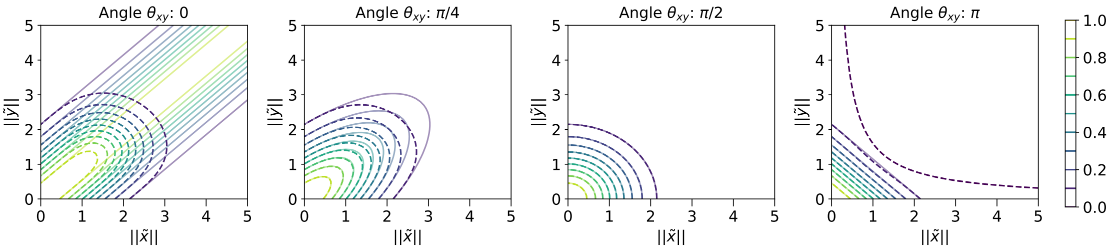

Fig. 1. $k_p(\boldsymbol{x}, \boldsymbol{y})$ (dashed) vs. $k(\boldsymbol{x}, \boldsymbol{y})$ (transparent and solid) for fixed $p = 3$ and $\sigma^2 = l = 1$, where we use the relationship $\tilde{\boldsymbol{x}}^\top \tilde{\boldsymbol{y}} = ||\tilde{\boldsymbol{x}}|| \, ||\tilde{\boldsymbol{y}}|| \cos(\theta_{xy})$.

can see that $k_p(\boldsymbol{x}, \boldsymbol{y})$ approaches zero with increasing $||\tilde{\boldsymbol{x}}||^2 + ||\tilde{\boldsymbol{y}}||^2$ regardless of θ_{xy}. This deteriorates the approximation quality, in particular as θ_{xy} goes to zero. This development accelerates when choosing a shorter length scale l. A consequence of this pathology is that the GPR predictive means and variances in Eq. (1) collapse to zero for test points $\tilde{\boldsymbol{x}^*}$ with large $||\tilde{\boldsymbol{x}}^*||$ since $k_p(\boldsymbol{x}^*, \boldsymbol{x})$ goes to zero for any $\boldsymbol{x} \in \mathbb{R}^d$ in this case. This effect is shown in the middle plot of Fig. (2). We will discuss this example in greater detail in Section 4.

A similar pathology was identified for the *Relevance Vector Machine* [18] that was solved by adding new basis functions centered on the test points [15]. In the following, we present a similar strategy by centering our entire approximation around individual test points.

3.2. Curing the pathology for GP regression

We will now exploit a property of the Gaussian kernel that allows us to cure the aforementioned pathology. The Gaussian kernel is shift-invariant, i.e., $k(\boldsymbol{x} + \boldsymbol{\delta}, \boldsymbol{y} + \boldsymbol{\delta}) = k(\boldsymbol{x}, \boldsymbol{y})$ for any $\boldsymbol{\delta} \in \mathbb{R}^d$, because $||(\boldsymbol{x} + \boldsymbol{\delta}) - (\boldsymbol{y} + \boldsymbol{\delta})|| = ||\boldsymbol{x} - \boldsymbol{y}||$. Thus, when making a prediction at a test input $\boldsymbol{x}_*$, one can subtract $\boldsymbol{x}_*$ from all inputs used in Eq. (1) without changing the result of the prediction. More specifically, the values μ_* and σ_*^2 in Eq. (1) do not change if we substitute $k(\boldsymbol{x}, \boldsymbol{y})$ by $k(\boldsymbol{x} - \boldsymbol{x}_*, \boldsymbol{y} - \boldsymbol{x}_*)$ for the computation of $\boldsymbol{k}_{\mathbf{f}*}, \boldsymbol{K}_{\mathbf{ff}}$ and k_{**}.

However, the approximate kernel $\hat{k}_p$ (6) is greatly affected by this change. To see this, we define:

$$\hat{k}_p^*(\boldsymbol{x}, \boldsymbol{y}) := \hat{k}_p(\boldsymbol{x} - \boldsymbol{x}_*, \boldsymbol{y} - \boldsymbol{x}_*) = \sigma^2 \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x}^*||^2}{2l^2}\right) \exp\left(-\frac{||\boldsymbol{y} - \boldsymbol{x}^*||^2}{2l^2}\right) \sum_{n=1}^{p} \frac{1}{n!} \Phi_n\left(\frac{\boldsymbol{x} - \boldsymbol{x}^*}{l}\right)^\top \Phi_n\left(\frac{\boldsymbol{y} - \boldsymbol{x}^*}{l}\right), \tag{7}$$

where $\mathbb{E}[\hat{k}_p^*(\boldsymbol{x}, \boldsymbol{y})] = k_p(\boldsymbol{x} - \boldsymbol{x}^*, \boldsymbol{y} - \boldsymbol{x}^*) =: k_p^*(\boldsymbol{x}, \boldsymbol{y})$. As all $\{\Phi_n\}_{n=1}^p$ are sampled independently, we have

$$\mathbb{V}[\hat{k}_p^*(\boldsymbol{x}, \boldsymbol{y})] = \mathbb{V}[\hat{k}_p(\boldsymbol{x} - \boldsymbol{x}^*, \boldsymbol{y} - \boldsymbol{x}^*)] \propto \sum_{n=1}^{p} \left(\frac{1}{n!}\right)^2 \mathbb{V}\left[\Phi_n\left(\frac{\boldsymbol{x} - \boldsymbol{x}^*}{l}\right)^\top \Phi_n\left(\frac{\boldsymbol{y} - \boldsymbol{x}^*}{l}\right)\right], \tag{8}$$

where the variance is with respect to the random feature distribution. When setting $\boldsymbol{x} = \boldsymbol{x}^*$ or $\boldsymbol{y} = \boldsymbol{x}^*$, the variance terms in Eq. (8) become *zero* for any of the polynomial sketches presented in 2.3 as can be seen from Table (1). $\hat{k}_p^*(\boldsymbol{x}^*, \boldsymbol{y}), \hat{k}_p^*(\boldsymbol{x}, \boldsymbol{x}^*)$ and $\hat{k}_p^*(\boldsymbol{x}^*, \boldsymbol{x}^*)$ thus become *deterministic*.

We further have $\hat{k}_p^*(\boldsymbol{x}^*, \boldsymbol{y}) = \sigma^2 \exp(-||\boldsymbol{x}^* - \boldsymbol{y}||^2/(2l)^2)$, $\hat{k}_p^*(\boldsymbol{x}, \boldsymbol{x}^*) = \sigma^2 \exp(-||\boldsymbol{x} - \boldsymbol{x}^*||^2/(2l)^2)$ and $\hat{k}_p^*(\boldsymbol{x}^*, \boldsymbol{x}^*) = \sigma^2$ which are all equal to the exact kernel k evaluated at these points. Therefore, $\boldsymbol{k}_{\mathbf{f}*}$ and k_{**} in Eq. (1) become exact for our Maclaurin approximation. $\boldsymbol{K}_{\mathbf{ff}}$ unfortunately remains affected by the vanishing approximate kernels described by Theorem (3.1) and by non-zero random feature variances. A crucial advantage of using $\hat{k}_p^*$ instead of $\hat{k}_p$ is that the GP predictive distribution (1) does not collapse to zero anymore as $||\tilde{\boldsymbol{x}}^*||$ grows. We illustrate this on the following synthetic example data set. As it is one-dimensional, we stick to the deterministic feature map in Eq. (4) for now.

Approximating the sinc function. We draw 50 noisy observations $\{y_i\}_{i=1}^{50}$ with $y_i = sinc(5x_i) + \epsilon_i$ and $\epsilon_i \overset{i.i.d.}{\sim} \mathcal{N}(0, \sigma^2_{\text{noise}} = 0.01)$, where $\{x_i\}_{i=1}^{50}$ are sampled independently and uniformly from the interval $[-1.5, 1.5]$. We then fit a reference GPR on this data set using the Gaussian kernel (3), where the hyperparameters l and $\sigma^2 > 0$ are found through a gradient based optimization of the log marginal likelihood [14, Chapter 5.4]. The length scale found for the reference GPR

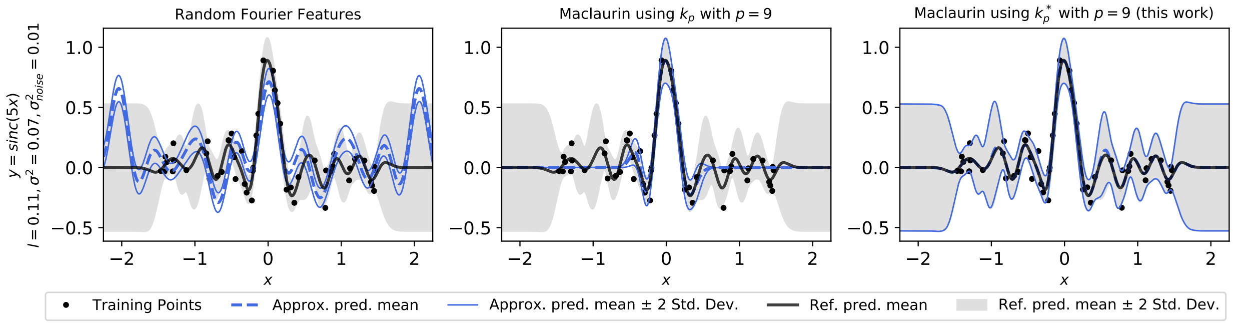

Fig. 2. Approximating the predictive distribution of a reference GPR using different approximation methods and $D = 10$.

is $l = 0.11$ and is rather short compared to 0.92, which is the median pairwise Euclidean distance of the training data, a standard heuristic for choosing the length scale without optimization [7]. This reflects the frequent oscillations of the function $\text{sinc}(5x)$. We show the reference GPR along with three different approximation schemes in Fig. (2). The baseline Random Fourier features (RFF) [13] (left) struggles to recover the reference GPR for $D = 10$ random features. The dimension D of the feature map (4) is equal to $p + 1$ for the Maclaurin approximation. So we chose $p = 9$ for a fair comparison against RFF.

The Maclaurin approach using k_p^* (right) gives the best approximations while the predictive distribution of the one using k_p (middle) collapses to zero very quickly at points $x^* \in \mathbb{R}$ away from zero. This is because $\|\tilde{x}^*\|$ becomes large very quickly and k_p vanishes when being evaluated at these points. For values x^* far away from the training data, the GP predictor (1) using k_p^* even recovers the GPR prior distribution as desired, which can be explained as follows. When x^* is far from the training data, $\boldsymbol{k}_{\mathbf{f}*}$ becomes zero. Then $\mu_* = 0$ and $\sigma_*^2 = k_{**} = \sigma^2$ in Eq. (1). Since k_{**} and $\boldsymbol{k}_{\mathbf{f}*}$ in Eq. (1) are accurate when using k_p^* as explained earlier, the convergence to the prior is kept for k_p^*.

3.3. Reducing computational costs through clustering

A caveat of using the approximate kernel $\hat{k}_p^*(\boldsymbol{x}, \boldsymbol{y})$ (7) described in Section 3.2 is that we need to recompute the GPR predictor (1) for every test point $\boldsymbol{x}^*$ separately, which becomes expensive when many test points need to be predicted, even when using the featurized version (2) of the predictor. We denote the number of test inputs by N_*. A direct computation of Eq. (2) now costs a total of $O(N_* ND^2)$ for all test points.

The problem is "embarrassingly parallel" and the computation of Eq. (2) for every $\boldsymbol{x}^*$ could be easily distributed on a cluster of compute nodes or parallelized using a single GPU, e.g., using JAX [2]. However, we propose a different approach here that requires no separate training at test time while staying as close as possible to the training time of $O(ND^2)$ as is generally desired for random feature approximations.

Our approach is to cluster the *training* data into N_C clusters and to use the centroids of these clusters as *pseudo test inputs*. We choose a farthest point clustering for simplicity as it does not require convergence verification and determines the number of clusters using a threshold $\theta > 0$, but any clustering algorithm can be used instead (even a random selection of training points). As the centroids are known during training, we can pretrain a set of N_C predictors using Eq. (2) and assign a new test point $\boldsymbol{x}^*$ to the closest centroid at prediction time. We summarize the complete procedure in Alg. (1). The computational cost is now $O(N_C ND^2)$ and is thus much lower than $O(N_* ND^2)$ if $N_C \ll N_*$.

4. Empirical evaluation on real-world data

In this section, we evaluate our proposed method in Section 3.2 on real-world data of different dimensions. Since $d > 2$ holds in all examples, we employ the polynomial sketches (5) as presented in Section 2.3. As for the synthetic example in Fig. (2), we use the Gaussian kernel with hyperparameters l and $\sigma^2 > 0$ that are found through gradient based optimization of the log marginal likelihood of a reference GPR, along with σ^2_{noise}.

Algorithm 1: Precomputing Localized Maclaurin Features for the Gaussian Kernel

```
Input: Hyperparameters $l, \sigma^2, \sigma^2_{\text{noise}} > 0$; training data $\{\boldsymbol{x}_i\}_{i=1}^{N}$; test data $\{\boldsymbol{x}_i^*\}_{i=1}^{N_*}$; number of features $D \geq 1$;
// 1) Training
if $d > 1$ then // We use random features
    Center the training data by subtracting the training mean ;
    Find $(p^*, D_1, \ldots, D_{p^*})$, the optimal feature allocation for the random Maclaurin method (see Section 2.3) ;
end
$C := \{(\sum_{i=1}^{N} \boldsymbol{x}_i)/N\}$ // Initialize centroids with training mean ;
while True do // Farthest point clustering
    Let $\delta_i = \min\{\|\boldsymbol{x}_i - \boldsymbol{c}\| \mid \boldsymbol{c} \in C\}$ for $i = 1, \ldots, N$ ;
    if $\max\{\delta_i\}_{i=1}^{N} < \theta$ then // Training Max.-Min.-Distance is below threshold $\theta$
        break;
    end
    Add $\boldsymbol{x}_i$ with $i = \arg\max_i \{\delta_i\}_{i=1}^{N}$ to $C$ ;
    Precompute $\boldsymbol{A}_c^{-1} := \left(\Phi(\boldsymbol{X} - \boldsymbol{c})^\top \Phi(\boldsymbol{X} - \boldsymbol{c})/\sigma^2_{\text{noise}} + \boldsymbol{I}\right)^{-1}$ with $\Phi$ defined in Eq. (4) ;
    // Use explicit features for $d = 1$, else use polynomial sketches (5) in Eq. (4)
end
forall $\{\boldsymbol{x}_i^*\}_{i=1}^{N_*}$ do // 2) Inference
    Assign $\boldsymbol{x}_i^*$ to closest centroid $\boldsymbol{c} \in C$ ;
    Compute $\mu_*$ and $\sigma_*^2$ in Eq. (2) by setting $\boldsymbol{A}^{-1} = \boldsymbol{A}_c^{-1}$, $\boldsymbol{x}^* = \boldsymbol{x}_i^* - \boldsymbol{c}$ and $\boldsymbol{X} = \boldsymbol{X} - \boldsymbol{c}$ ;
end
```

We compare our method against a random Fourier features baseline [13] as well as its structured extension [23] when the data is sufficiently high dimensional[2]. We also add the vanilla optimized Maclaurin method (Section 2.3) to this comparison. It is equivalent to Alg. (1) using only a single cluster with its centroid being the training mean. We measure the approximation quality with respect to the reference GPR using the Kullback-Leibler (KL) divergence [14, Chapter A.5] between the predictive means and variances in Eq. (1) and Eq. (2). We measure downstream regression performance using the root mean squared error (RMSE).

4.1. UK apartment price data

We downloaded the monthly property sales data for England and Wales from the HM land registry[3]. We filtered for sold apartments for the month of January 2022 leading to a data set with 24 553 observations. Matching the post codes for each apartment sold with a database of latitudes and longitudes[4] allowed us to obtain a two-dimensional data set (latitude, longitude) that we could regress against the logarithm of the sales prices. We randomly split the data into 10 000 training points and kept the rest for testing.

In our first experiment, we aim to recover the reference GPR predictive distribution on a regular grid of latitudes (between +50° and +55°) and longitudes (between −6° and +2°) of size 100 by 100. Fig. (3) shows the results of this experiment. As for the sinc-example in Fig. (2), random Fourier features struggle to recover the predictive distribution using $D = 100$ random features and the vanilla Maclaurin method using $\hat{k}_p$ (6) suffers from vanishing kernels due to the short (compared to the scaling of the data) length scale of $l = 0.25$. Our proposed kernel $\hat{k}_p^*$ (7) improves predictions considerably leading to the lowest KL divergence with respect to the reference GP predictive distribution. It also converges to the prior for test points far from the training data.

[2] Otherwise, the structured random Fourier features induce a large bias.

[3] https://www.gov.uk/government/statistical-data-sets/price-paid-data-downloads

[4] https://www.freemaptools.com/download-uk-postcode-lat-lng.htm

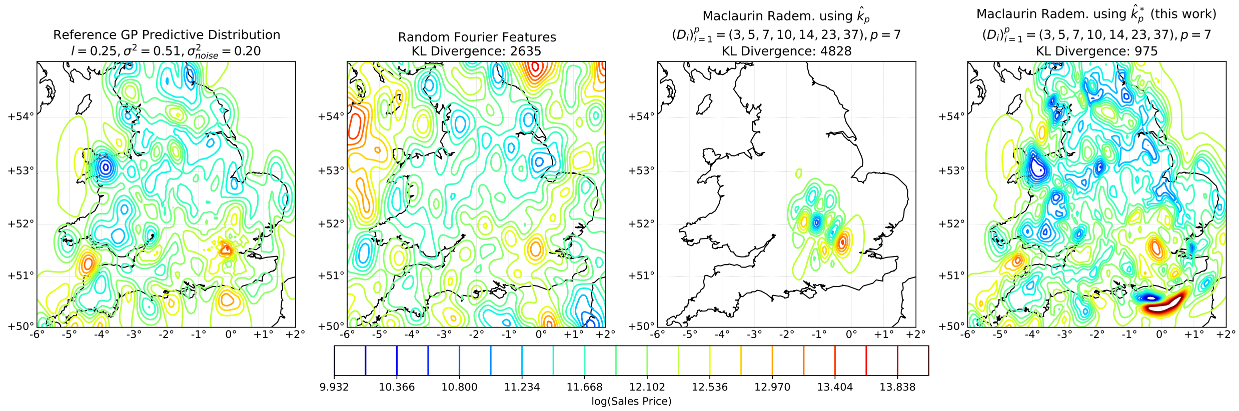
Reference GP Predictive Distribution
$l = 0.25, \sigma^2 = 0.51, \sigma^2_{noise} = 0.20$
Random Fourier Features
KL Divergence: 2635
Maclaurin Radem. using $\hat{k}_p$
$(D_i)_{i=1}^p = (3, 5, 7, 10, 14, 23, 37), p = 7$
KL Divergence: 4828
Maclaurin Radem. using $\hat{k}_p^*$ (this work)
$(D_i)_{i=1}^p = (3, 5, 7, 10, 14, 23, 37), p = 7$
KL Divergence: 975
+54°
+53°
+52°
+51°
+50°
-6° -5° -4° -3° -2° -1° 0° +1° +2°
9.932
10.366
10.800
11.234
11.668
12.102
12.536
12.970
13.404
13.838
log(Sales Price)

Fig. 3. Approximating the predictive mean of a reference GP using different approximation methods and $D = 100$. KL Divergences also include predictive variances. The Maclaurin method using $\hat{k}_p$ makes predictions centered around the mean of the training data (close to London).

In our second experiment, we evaluate the use of Alg. (1) to precompute the matrix inversion in Eq. (2) on a set of pseudo test inputs. This time we report results on the left-out test data instead of a regular grid. The left part of Fig. (4) shows these results. We can see that the KL divergence falls off considerably (top plot) as we add more clusters until reaching 57 clusters. From then on the KL divergence remains roughly the same indicating that 57 clusters give a good trade-off between efficiency and performance. In the bottom plot we show a comparison of RMSE values for these 57 clusters, where the Maclaurin method outperforms random Fourier features, in particular for small D.

4.2. UCI data sets: Yacht and kin8nm

In the following, we repeat the evaluation of Alg. (1) for two higher dimensional data sets that are taken from the UCI machine learning repository [4] in Fig. (4). We obtain very similar results for the UCI Yacht data set as for the UK apartment price data set. Adding more clusters gives large gains initially but these diminish when setting $\theta > 2.0l$ in Alg. (1) determining a good trade-off between performance and computational cost. This time we included structured orthogonal random Fourier features [23] that are also outperformed by the Maclaurin method.

For the UCI kin8nm data set, results look quite different. Adding more clusters *increases* the KL divergence towards the reference GP predictor, which is why only a single centroid, the mean of the training data, is chosen for the RMSE comparison in the plot below. This corresponds to the vanilla optimized Maclaurin method (Section 2.3).

We explain this observation as follows. The length scales obtained for the sinc example, the UK apartment price data and for UCI Yacht are short. For the UCI Yacht data set it is 0.32, i.e., much less than 3.28, the median pairwise Euclidean distance of the training data, indicating that the data is fit by a reference GP of high frequency. For kin8nm the length scale is 2.15 compared to 3.93 (median heuristic) indicating a much smoother GP than the ones before.

In this case, the values of $\boldsymbol{k}_{\mathbf{f}*}$ and k_{**} in Eq. (1) are less affected by the vanishing kernels (Theorem 3.1) due to a longer length scale. However, the approximation of $\boldsymbol{K}_{\mathbf{ff}}$ in Eq. (1) is more accurate when using the vanilla Maclaurin approximation (6) because the data is centered around the training mean. This shows that the clusters need to be chosen depending on the smoothness of the target GP. In this work, we have provided a *generalization* of the vanilla Maclaurin method that (with an appropriate choice of clusters) can fit both, high and low-frequency data.

5. Conclusion

We have identified a major pathology when using Maclaurin-based approximations such as [3, 20] for the Gaussian kernel. We have further presented an extension of the optimized Maclaurin method [20] that overcomes this problem and makes it applicable to high-frequency data. The clustering method in Alg. (1) seems to have a strong impact on predictive performance. Future work should investigate on optimal clustering schemes that automatically adapt to the frequency of the target function. It would further be interesting to combine the advantages of random Fourier features

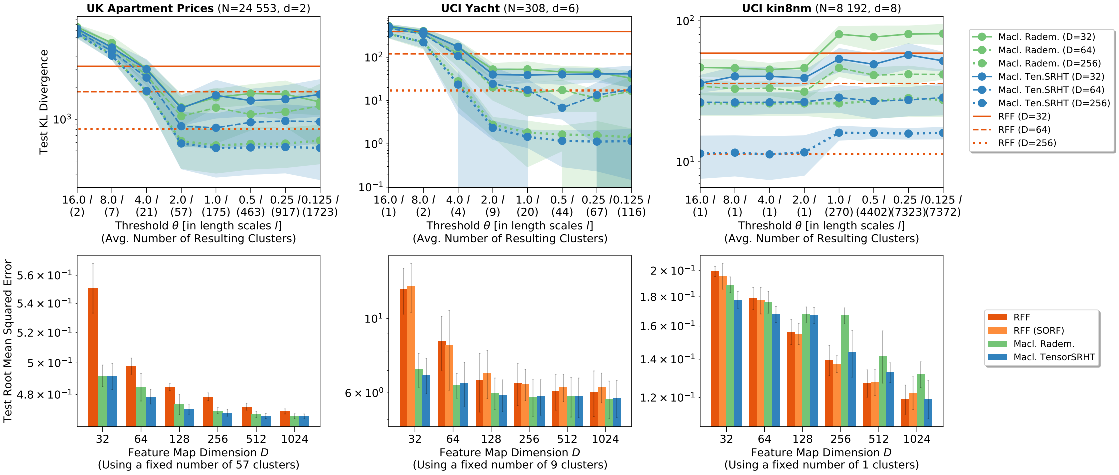

Fig. 4. (Top) Test KL Divergence for a given number of clusters obtained from the training data using Alg. (1) with threshold θ. (Bottom) RMSE over feature map dimension D for a fixed number of clusters.

and polynomial sketches, as both approximations have zero variances in different regimes (equal inputs for RFF and orthogonal inputs for Maclaurin).

Acknowledgements. We thank Sagar Arora for helpful discussions. MF gratefully acknowledges support from the AXA Research Fund and the Agence Nationale de la Recherche (ANR-18-CE46-0002 and ANR-19-P3IA-0002).

Appendix A. TensorSRHT: A structured polynomial sketch

Algorithm 2: TensorSRHT according to [20]

```
Result: A feature map $\phi_n(\boldsymbol{x})$
Pad $\boldsymbol{x}$ with zeros so that $d$ becomes a power of 2 and let $\boldsymbol{H}_d \in \{1,-1\}^{d\times d}$ be the unnormalized
  Walsh-Hadamard matrix [5]; Let $B = \left\lceil \frac{D}{d} \right\rceil$ be the number of stacked projection blocks ;
forall $b \in \{1,\dots,B\}$ do
    forall $i \in \{1,\dots,n\}$ do
        Generate a random vector $\mathbf{d}_i = (d_{i,1},\dots,d_{i,d})^\top \in \mathbb{R}^d$ as $d_{i,1},\dots,d_{i,d} \overset{i.i.d.}{\sim} \text{Unif}(\{1,-1\})$;
        Compute $\phi^{b,i}(\boldsymbol{x}) := \boldsymbol{H}_d(\boldsymbol{d}_i \odot \boldsymbol{x})$ using the FWHT [5] and shuffle the elements of $\phi^{b,i}(\boldsymbol{x})$ randomly;
    end
    Compute $\phi^b(\boldsymbol{x}) := (\phi^{b,1}(\boldsymbol{x}) \odot \cdots \odot \phi^{b,n}(\boldsymbol{x}))/\sqrt{D}$ ;
end
Concatenate the elements of $\phi^1(\boldsymbol{x}),\dots,\phi^B(\boldsymbol{x})$ to yield a single vector $\phi_n(\boldsymbol{x})$ and keep the first $D$ entries ;
```

Appendix B. Proof of Theorem 3.1

Proof. We start by deriving an upper bound for $|k_p(\boldsymbol{x},\boldsymbol{y})|$. We leave out the length scale here for ease of notation.

$$|k_p(\boldsymbol{x},\boldsymbol{y})| = \sigma^2 \exp(-(||\boldsymbol{x}||^2 + ||\boldsymbol{y}||^2)/2) \left| \sum_{n=0}^{p} 1/n!(\boldsymbol{x}^\top \boldsymbol{y})^n \right| \leq \sigma^2 \exp(-(||\boldsymbol{x}||^2 + ||\boldsymbol{y}||^2)/2) \sum_{n=0}^{p} 1/n!(||\boldsymbol{x}||\,||\boldsymbol{y}||)^n$$

Next, we notice that $||\boldsymbol{x} - \boldsymbol{y}||^2 = ||\boldsymbol{x}||^2 + ||\boldsymbol{y}||^2 - 2\boldsymbol{x}^\top \boldsymbol{y} \geq 0$ for any $\boldsymbol{x}, \boldsymbol{y}$. Thus, we can choose them to be parallel. So $||\boldsymbol{x}|| \, ||\boldsymbol{y}|| \leq (||\boldsymbol{x}||^2 + ||\boldsymbol{y}||^2)/2$. From this inequality, it follows

$$\sum_{n=0}^{p} 1/n!(||\boldsymbol{x}|| \, ||\boldsymbol{y}||)^n \leq \sum_{n=0}^{p} 1/n!((||\boldsymbol{x}||^2 + ||\boldsymbol{y}||^2)/2)^n \leq \exp((||\boldsymbol{x}||^2 + ||\boldsymbol{y}||^2)/2).$$

Now, the gap $\exp((||\boldsymbol{x}||^2+||\boldsymbol{y}||^2)/2) - \sum_{n=0}^{p} 1/n!((||\boldsymbol{x}||^2+||\boldsymbol{y}||^2)/2)^n = \sum_{n=p+1}^{\infty} 1/n!((||\boldsymbol{x}||^2+||\boldsymbol{y}||^2)/2)^n$ increases as $||\boldsymbol{x}||^2+||\boldsymbol{y}||^2$ increases, which must decreases the ratio $\sum_{n=0}^{p} 1/n!(||\boldsymbol{x}|| \, ||\boldsymbol{y}||)^n / \exp((||\boldsymbol{x}||^2 + ||\boldsymbol{y}||^2)/2)$ and thus the upper bound of $|k_p(\boldsymbol{x}, \boldsymbol{y})|$ goes to zero as $||\boldsymbol{x}||^2 + ||\boldsymbol{y}||^2$ increases.

Next, we look at the error $|k(\boldsymbol{x}, \boldsymbol{y}) - k_p(\boldsymbol{x}, \boldsymbol{y})| = \sigma^2 \exp(-(||\boldsymbol{x}||^2 + ||\boldsymbol{y}||^2)/2) \left|\sum_{n=p+1}^{\infty} 1/n!(\boldsymbol{x}^\top \boldsymbol{y})^n\right|$. If the angle between $\boldsymbol{x}$ and $\boldsymbol{y}$ is zero such that $\boldsymbol{x}^\top \boldsymbol{y} = ||\boldsymbol{x}|| \, ||\boldsymbol{y}||$, all addends in the infinite sum are maximized. The error thus becomes the largest. The error is zero when they are orthogonal. □

References

[1] Ahle, T.D., Kapralov, M., Knudsen, J.B.T., Pagh, R., Velingker, A., Woodruff, D.P., Zandieh, A., 2020. Oblivious sketching of high-degree polynomial kernels, in: Proceedings of the Thirty-First Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics. p. 141–160.

[2] Bradbury, J., Frostig, R., Hawkins, P., Johnson, M.J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., Zhang, Q., 2018. JAX: composable transformations of Python+NumPy programs. URL: http://github.com/google/jax.

[3] Cotter, A., Keshet, J., Srebro, N., 2011. Explicit approximations of the gaussian kernel. CoRR abs/1109.4603. arXiv:1109.4603.

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

[5] Fino, B.J., Algazi, V.R., 1976. Unified matrix treatment of the fast walsh-hadamard transform. IEEE Transactions on Computers 25, 1142–1146.

[6] Floudas, C.A., Pardalos, P.M. (Eds.), 2009. Encyclopedia of Optimization, Second Edition. Springer.

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

[8] Hamid, R., Xiao, Y., Gittens, A., DeCoste, D., 2014. Compact random feature maps, in: Proceedings of the 31th International Conference on Machine Learning, PMLR. pp. 19–27.

[9] Hensman, J., Durrande, N., Solin, A., 2018. Variational Fourier features for Gaussian processes. Journal of Machine Learning Research 18, 1–52.

[10] Kar, P., Karnick, H., 2012. Random feature maps for dot product kernels, in: Proceedings of the Fifteenth International Conference on Artificial Intelligence and Statistics, JMLR. pp. 583–591.

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

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

[13] Rahimi, A., Recht, B., 2007. Random features for large-scale kernel machines, in: Advances in Neural Information Processing Systems 20, Curran Associates Inc.. pp. 1177–1184.

[14] Rasmussen, C., Williams, C., 2006. Gaussian Processes for Machine Learning. MIT Press.

[15] Rasmussen, C.E., Quiñonero Candela, J., 2005. Healing the relevance vector machine through augmentation, in: Proceedings of the 22nd International Conference on Machine Learning, PMLR. p. 689–696.

[16] Rudi, A., Carratino, L., Rosasco, L., 2017. Falkon: An optimal large scale kernel method, in: Advances in Neural Information Processing Systems, Curran Associates, Inc.

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

[18] Tipping, M.E., 1999. The relevance vector machine, in: Advances in Neural Information Processing Systems 12, Curran Associates, Inc.. p. 652–658.

[19] Titsias, M., 2009. Variational learning of inducing variables in sparse Gaussian processes, in: Proceedings of the Twelfth International Conference on Artificial Intelligence and Statistics, JMLR. pp. 567–574.

[20] Wacker, J., Kanagawa, M., Filippone, M., 2022a. Improved random features for dot product kernels. arXiv preprint arXiv:2201.08712 .

[21] Wacker, J., Ohana, R., Filippone, M., 2022b. Complex-to-real random features for polynomial kernels. arXiv preprint arXiv:2202.02031 .

[22] Williams, C.K., Seeger, M., 2000. Using the Nyström method to speed up kernel machines, in: Advances in Neural Information Processing Systems 13, Curran Associates, Inc.. pp. 682–688.

[23] Yu, F.X., Suresh, A.T., Choromanski, K., Holtmann-Rice, D., Kumar, S., 2016. Orthogonal random features, in: Advances in Neural Information Processing Systems 30, Curran Associates Inc.. p. 1983–1991.