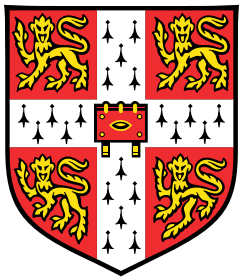
Efficient Gaussian Processes for data-driven decision making



Vincent Dutordoir

Department of Engineering
University of Cambridge

Report submitted to be registered for the PhD Degree
First-Year-Report

Jesus College

August 2021

Abstract

This is where you write your abstract ...

Table of contents

Chapter 1

Introduction

Using past experience to update ones behaviour is what differentiates intelligent beings from other creatures in our world. To artificially create said systems, that can learn from data acquired through experience, is the crowning goal of the field of Artificial Intelligence (AI) and Machine Learning (ML). Realising this goal will have a large impact on the world and society we live in, as it will free up humans from tasks that require cognition, like driving cars, booking appointments, and interpreting and acting on medical records, to give a few examples. Though, arguably, the field has still a long way to go before fulfilling its full potential.

An elegant approach to formalise the concept of learning through acquired experience is that of *Bayesian learning*. In this framework, one specifies beliefs over quantities of interest, and updates them after observing data. The beliefs are represented using probability distributions: the *prior* describes what is known about the quantity before any data is observed, and the so-called *likelihood* describes the relation between the observed data and the quantity of interest. The framework provides a recipe for obtaining an updated belief over the quantity of interest after data has been observed. This distribution is known as the *posterior*. Bayesian learning makes decisions atop these beliefs, and uses the posterior to reason about the optimality of a decision or the cost associated to a certain action.

The performance of a decision-making system will depend on the speed at which it can learn and the optimality of the decisions made—quantified by the *regret*. It can be shown that, under certain assumptions, Bayesian learning gives rise to the optimal regret. However, such excellence comes at a high computational cost, which in many scenarios renders Bayesian learning—in its purest form—unpractical. Fortunately, the literature has proposed many approximate methods which have lightened the computational complexity of the Bayesian paradigm.

Unquestionably, (approximate) models for decision-making systems need to deal with *uncertainty*. They need to be able to quantify what is known, and what is not known. The importance of quantifying uncertainty for decision-making systems becomes clear from the many sources it can stem from. For example, there can be multiple different settings of a model that explain the data, so one needs to be uncertain about the setting that actually generated it.

One also needs to be uncertain about the model itself, as most probably the model at hand is a simplification of the real-world process. Moreover, the environment in which the system operates may also be inherently uncertain, in which case even an infinite amount of data (i.e. experience) would not make the system any smarter (e.g., trying to predict the outcome of rolling a fair dice).

In my opinion, Gaussian processes are the perfect compromise between computational complexity and Bayesian rigour. Their non-parametric nature makes them complex enough to model a wide range of problems, while their kernel formulation makes them applicable to many different domains: graphs, vectors, images, etc. Instead of representing probability distributions on weights, Gaussian processes can be used to represent uncertainty directly on the function that the weights represents. The *function-space* view of Gaussian processes makes them more amenable to mathematical analysis, which in turn leads to strong guarantees on the future performance of the system and overall robustness. This may be necessary for deploying these systems in critical applications. For these reasons, I believe, studying Gaussian processes is very worthwhile.

In this report, I present two related pieces of novel research in the domain of approximate Bayesian inference for Gaussian process models. In chapter 3, I introduce an interdomain inducing variable approach that speeds up inference and prediction in Gaussian processes by two orders of magnitude. An important building block for this work are spherical harmonics, for which we developed an efficient algorithm for their evaluation in high dimensions. In section 3.4, I present this algorithm and its theoretical foundation. In chapter 4 we marry the strengths of deep neural networks and deep Gaussian processes by establishing an equivalence between the forward passes of both models. This results in models that can either be seen as neural networks with improved uncertainty prediction or deep Gaussian processes with increased prediction accuracy. The final part of this report, chapter 5, is devoted to future research. Before commencing, we start with covering the necessary theoretical background in chapter 2.

The material presented in chapters 3 and 4 is either published or is currently under review:

1. Vincent Dutordoir, Nicolas Durrande, and James Hensman [2020a]. "Sparse Gaussian Processes with Spherical Harmonic Features". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*

2. Vincent Dutordoir, James Hensman, Mark van der Wilk, Carl Henrik Ek, Zoubin Ghahramani, and Nicolas Durrande [2021a]. "Deep Neural Networks as Point Estimate for Deep Gaussian Processes". In: *submission to NeurIPS*

3. Vincent Dutordoir, Hugh Salimbeni, Eric Hambro, John McLeod, Felix Leibfried, Artem Artemev, Mark van der Wilk, James Hensman, Marc P Deisenroth, and ST John [2021b]. "GPflux: A Library for Deep Gaussian Processes". In: *Proceedings of the 3th International Conference on Probabilistic Programming (2021)*

Chapter 2

Theoretical Framework

This chapter discusses Gaussian processes (GP) and deep Gaussian processes (DGPs), the composite model obtained by stacking multiple GP models on top of each other. We also review how to perform approximate Bayesian inference in these models, with a particular attention to Variational Inference. We also cover the theory of positive definite kernels and the Reproducing Kernel Hilbert Spaces (RKHS) they characterise.

2.1 Gaussian Processes

Gaussian processes (GPs) [Rasmussen and Williams, 2006] are non-parametric distributions over functions similar to Bayesian Neural Networks (BNNs). The core difference is that neural networks represent distributions over functions through distributions on weights, while a Gaussian process specifies a distribution on function values at a collection of input locations. This representation allows us to use an infinite number of basis functions, while still enables Bayesian inference [Neal, 1995].

Following from the Kolmogorov extension theorem, we can construct a real-valued stochastic process (i.e. function) on a non-empty set $\mathcal{X}$, $f : \mathcal{X} \to \mathbb{R}$, if there exists on all finite subsets $\{x_1, \dots x_N\} \subset \mathcal{X}$, a *consistent* collection of finite-dimensional marginal distributions over $f(\{x_1, \dots, x_n\})$. For a Gaussian process, in particular, the marginal distribution over every finite-dimensional subset is given by a multivariate normal distribution. This implies that, in order to fully specify a Gaussian process, it suffice to only define the mean and covariance (kernel) function because they are the sufficient statistics for every finite-dimensional marginal distribution. We can therefore denote the GP as

$$f \sim \mathcal{GP}(\mu, k), \tag{2.1}$$

where $\mu : \mathcal{X} \to \mathbb{R}$ is the mean function, which encodes the expected value of f at every x, $\mu(x) = \mathbb{E}_f[f(x)]$, and $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the covariance (kernel) function that describes the

covariance between function values, $k(x, x') = \text{Cov}(f(x), f(x'))$. The covariance function has to be a symmetric, positive-definite function. The Gaussianity, and the fact that we can manipulate function values at some finite points of interest without taking the behaviour at any other points into account (the marginalisation property) make GPs particularly convenient to manipulate and use as priors over functions in Bayesian models —as we will show next.

Throughout this report, we will consider f to be the complete function, and intuitively manipulate it as an infinitely long vector. When necessary, we will append '$(\cdot)$' to f to highlight that it is indeed a function rather than a finite vector. Contrarily, $f(\boldsymbol{x}) \in \mathbb{R}^N$ denotes the function evaluated at a finite set of points $\boldsymbol{x} = \{x_1, \cdots, x_N\}$. $f^{\setminus \boldsymbol{x}}$ denotes another infinitely long vector similar to f but excluding $f(\boldsymbol{x})$. Intuitively, f can be partitioned into two groups: $f(\boldsymbol{x})$ and $f^{\setminus \boldsymbol{x}}$, with the following joint distribution

$$\begin{pmatrix} f^{\setminus \boldsymbol{x}}(\cdot) \\ f(\boldsymbol{x}) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu(\cdot) \\ \boldsymbol{\mu}_{\mathbf{f}} \end{pmatrix}, \begin{pmatrix} k(\cdot,\cdot) & \mathbf{k}_{\mathbf{f}\cdot}^\top \\ \mathbf{k}_{\mathbf{f}\cdot} & \mathbf{K}_{\mathbf{ff}} \end{pmatrix} \right) \tag{2.2}$$

where $[\boldsymbol{\mu}_{\mathbf{f}}]_i = \mu(x_i)$, $[\mathbf{K}_{\mathbf{ff}}]_{i,j} = k(x_i, x_j)$, $[\mathbf{k}_{\mathbf{f}\cdot}]_i = k(x_i, \cdot)$, and $\mu(\cdot)$ and $k(\cdot,\cdot)$ are the mean and covariance function from eq. (2.1). Following the marginalisation property, the distribution over $f(\boldsymbol{x})$ is given by

$$p(f(\boldsymbol{x})) = \int p(f)\, \mathrm{d}f^{\setminus \boldsymbol{x}} = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{f}}, \mathbf{K}_{\mathbf{ff}}). \tag{2.3}$$

We can now specify the GP at this finite set of points and use the rules of conditioning to obtain the GP elsewhere. Let $f(\boldsymbol{x}) = \mathbf{f}$, then the conditional distribution for $f^{\setminus \boldsymbol{x}}$ is given by another Gaussian process

$$p(f^{\setminus \boldsymbol{x}}(\cdot) \mid f(\boldsymbol{x}) = \mathbf{f}) = \mathcal{GP}\Big(\mu(\cdot) + \mathbf{k}_{\mathbf{f}\cdot}^\top \mathbf{K}_{\mathbf{ff}}^{-1}(\mathbf{f} - \boldsymbol{\mu}_{\mathbf{f}}), \quad k(\cdot,\cdot) - \mathbf{k}_{\mathbf{f}\cdot}^\top \mathbf{K}_{\mathbf{ff}}^{-1} \mathbf{k}_{\mathbf{f}\cdot}\Big), \tag{2.4}$$

The conditional distribution over the whole function $p(f(\cdot) \mid f(\boldsymbol{x}) = \mathbf{f})$ has the exact same form as in eq. (2.4). This is mathematically slightly confusing because the random variable $f(\boldsymbol{x})$ is included both on the left and right-hand-side of the conditioning, but the equation is technically correct, as discussed in Matthews et al. [2016], van der Wilk et al. [2020], and Leibfried et al. [2020].

2.1.1 The Beauty of Gaussian Process Regression: Exact Bayesian Inference

A defining advantages of GPs is that we can perform exact Bayesian inference in the case of regression. Assume a supervised learning setting where $x \in \mathcal{X}$ (typically, $\mathcal{X} = \mathbb{R}^d$) and $y \in \mathbb{R}$, and we are given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^N$ of input and corresponding output pairs. For convenience, we sometimes group the inputs in $\boldsymbol{x} = \{x_i\}_{i=1}^N$ into a single design matrix and outputs $\boldsymbol{y} = \{y_i\}_{i=1}^N$ into a vector. We further assume that the data is generated by an unknown function $f : \mathcal{X} \to \mathbb{R}$, and that the outputs are perturbed versions of functions evaluations at

the corresponding inputs: $y_i = f(x_i) + \epsilon_i$. In the case of regression we assume a Gaussian noise model $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$. We are interested in learning the function f that generated the data.

Following the Bayesian approach, we specify a *prior* over the parameters of interests, which in the case of GPs is the function itself. The prior is important because it characterises the search space over possible solutions for f. Through the prior, we can encode strong assumptions, such as linearity, differentiability, periodicity, etc. or any combination thereof. These strong inductive biases make it possible to generalise well from very limited data. Compared to Bayesian parametric models, it is much more convenient and intuitive to specify priors directly in *function-space*, rather than on the weights of a parametric model [Rasmussen and Williams, 2006].

In Gaussian process regression (GPR), we specify a GP prior over f, for which we assume without loss of generality a zero mean function:

$$f \sim \mathcal{GP}(0, k), \tag{2.5}$$

The likelihood, describing the relation between the quantity of interest f and the observed data, is given by

$$p(\boldsymbol{y} \mid f) = \prod_{i=1}^{N} p(y_i \mid f) = \prod_{i=1}^{N} \mathcal{N}\Big(y_i \mid f(x_i), \sigma^2\Big), \tag{2.6}$$

where we see that, conditioned on the GP, the likelihood factorises over datapoints. The posterior over the complete function f is another GP, which can be obtained using Bayes' rule:

$$p(f \mid \boldsymbol{y}) = \mathcal{GP}\Big(\mathbf{k}_{\mathbf{f}\cdot}^\top (\mathbf{K}_{\mathbf{ff}} + \sigma^2 \mathbf{I})^{-1} \boldsymbol{y}, \quad k(\cdot,\cdot) - \mathbf{k}_{\mathbf{f}\cdot}^\top (\mathbf{K}_{\mathbf{ff}} + \sigma^2 \mathbf{I})^{-1} \mathbf{k}_{\mathbf{f}\cdot}\Big). \tag{2.7}$$

Similarly, the marginal likelihood (model evidence) is also available in closed-form and obtained by marginalising over the prior:

$$p(\boldsymbol{y}) = \mathcal{N}\Big(\boldsymbol{y} \mid \mathbf{0}, \mathbf{K}_{\mathbf{ff}} + \sigma^2 \mathbf{I}\Big). \tag{2.8}$$

The availablility of the posterior and the marginal likelihood in analytic form makes GPs a unique tool for inferring unknown functions from data.

Despite the convenience of the analytic expressions, computing this exact posterior requires inverting the $N \times N$ matrix $\mathbf{K}_{\mathbf{ff}}$, which has a $\mathcal{O}(N^3)$ computational complexity and a $\mathcal{O}(N^2)$ memory footprint. Interestingly, the computational intractability in GPR models originates from the prior and *not* the complexity of marginalisation as is often the case for other Bayesian methods [TODO: CITE]. Indeed, these methods resort to MCMC to avoid computing intractable normalising constant, which is a problem that GPR does not encounter. Another shortcoming of GPR models is that there is no known analytical expression for the posterior distribution when the likelihood (eq. (2.6)) is not Gaussian, as encountered in classification for instance. In the next paragraph we discuss solutions to both problems.

2.2 Approximate Inference with Sparse Gaussian Processes

Sparse Gaussian processes combined with Variational Inference (VI) provide an elegant way to address these two shortcomings [Titsias, 2009; Hensman et al., 2013; 2015]. VI consists of introducing a distribution $q(f)$ that depends on some parameters, and finding the values of these parameters such that $q(f)$ gives the best possible approximation of the exact posterior $p(f \mid \boldsymbol{y})$. It is worth noting that there exists other approaches in the literature that address the same issues. In particular, Expectation Propagation (EP) [Minka, 2001; Bui et al., 2017] formulates, similar to VI, an approximation to the posterior but uses a different objective to optimise the approximation. Tangentially, other sparse GP methods (e.g., FITC) [Snelson and Ghahramani, 2005; Quiñonero-Candela and Rasmussen, 2005] approximate the model rather than the posterior. A downside of this is that the posteriors lose their non-parametric nature, which can be detrimental for the uncertainty estimates [Bauer et al., 2016]. In what follows we will focus on Sparse Gaussian processes with variational inference because of its general applicability and overall robust performance.

We first discuss the objective used in general variational inference before specifying our particular choice for $q(f)$. Let us therefore define $q(f)$ as the approximate to the posterior, then the idea is to optimise $q(f)$ so that the distance measured by the Kullback–Leibler (KL) divergence from $q(f)$ to $p(f \mid \boldsymbol{y})$ is minimal. Rewriting $p(f \mid \boldsymbol{y})$ using Bayes' rule, we obtain:

$$\mathrm{KL}[q(f) \,\|\, p(f \mid \boldsymbol{y})] = -\mathbb{E}_{q(f)}\left[\log \frac{p(\boldsymbol{y} \mid f)p(f)}{q(f)}\right] + \log p(\boldsymbol{y}). \tag{2.9}$$

Rearranging these terms yields:

$$\log p(\boldsymbol{y}) - \mathrm{KL}[q(f) \,\|\, p(f \mid \boldsymbol{y})] = \underbrace{\mathbb{E}_{q(f)}[\log p(\boldsymbol{y} \mid f)] - \mathrm{KL}[q(f) \,\|\, p(f)]}_{\triangleq \mathrm{ELBO}} \tag{2.10}$$

The r.h.s. of eq. (2.10) is known as the Evidence Lower BOund (ELBO). It is a lower bound to the log marginal likelihood ($\log p(\boldsymbol{y})$) because the KL between $q(f)$ and $p(\boldsymbol{y} \mid f)$ is always non-negative. Further, since $p(\boldsymbol{y})$ does not depend on the variational approximation, maximising the ELBO w.r.t. $q(f)$ is equivalent to minimsing the $\mathrm{KL}[q(f) \,\|\, p(f \mid \boldsymbol{y})]$. The maximum will be reached when $q(f) = p(f \mid \boldsymbol{y})$, in which case the KL will be zero and the ELBO equals the log evidence.

Intuitively, VI casts the problem of Bayesian inference, namely marginalisation, as an optimisation problem. The objective for the optimisation problem is the ELBO, which depends on the variational approximation, the prior and the data but, importantly, not the true posterior. This approach has several advantages. Firstly, optimisation, as opposed to marginalisation, is garantueed to converge —albeit possibly to a local optima. Secondly, VI provides a framework in which one can trade computational cost for accuracy: by expanding the family of approximating distributions we can only tighten the bound. An interesting example of this is importance

weighting, where in the limit of infinite compute the true posterior is part of the approximating family [Domke and Sheldon, 2018]. Finally, the bound and its derivates can be computed with stochastic estimations using the reparametrisation trick or score function estimators which enables big data settings.

So far we have discussed the bound in VI, but have left $q(f)$ unspecified, we now focus our attention to this. In general, we want the family of variational distributions to be flexible enough, such that some setting of the parameters can approximate the true posterior well, while also being computationally efficient and mathematically convenient to manipulate. We follow the approach proposed by Titsias [2009], which parameterises the approximation using a set of M pseudo inputs $\boldsymbol{z} = \{z_1, \dots, z_M\} \in \mathbb{R}^{M \times d}$ corresponding to M inducing variables $\boldsymbol{u} = f(\boldsymbol{z}) \in \mathbb{R}^M$. We choose to factorise the variational approximation as $q(f, \boldsymbol{u}) = q(\boldsymbol{u})p(f \mid f(\boldsymbol{z}) = \boldsymbol{u})$, where

$$p(f \mid f(\boldsymbol{z}) = \boldsymbol{u}) = \mathcal{GP}\Big(\mathbf{k}_{\boldsymbol{u}\cdot}^\top \mathbf{K}_{\boldsymbol{uu}}^{-1}\boldsymbol{u}, \quad k(\cdot,\cdot) - \mathbf{k}_{\boldsymbol{u}\cdot}^\top \mathbf{K}_{\boldsymbol{uu}}^{-1}\mathbf{k}_{\boldsymbol{u}\cdot}\Big), \tag{2.11}$$

with $[\mathbf{K}_{\boldsymbol{uu}}]_{i,j} = k(z_i, z_j)$, $[\mathbf{k}_{\boldsymbol{u}\cdot}]_i = k(z_i, \cdot)$. We specify a Gaussian distribution for the variational posterior over the inducing variables with a freely parameterised mean and covariance

$$q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m}, \mathbf{S}) \tag{2.12}$$

such that the overall approximate posterior is given by another GP

$$q(f) = \int_{\boldsymbol{u}} q(f, \boldsymbol{u})\, \mathrm{d}\boldsymbol{u} = \mathcal{GP}\Big(\mathbf{k}_{\boldsymbol{u}\cdot}^\top \mathbf{K}_{\boldsymbol{uu}}^{-1}\boldsymbol{m}, \quad k(\cdot,\cdot) - \mathbf{k}_{\boldsymbol{u}\cdot}^\top \mathbf{K}_{\boldsymbol{uu}}^{-1}(\mathbf{K}_{\boldsymbol{uu}} + \mathbf{S})\mathbf{K}_{\boldsymbol{uu}}^{-1}\mathbf{k}_{\boldsymbol{u}\cdot}\Big). \tag{2.13}$$

as a result of the conjugacy of the variational posterior $q(\boldsymbol{u})$ and the conditional. The approximation admits several interesting properties. Firstly, in the case of a Gaussian likelihood and the number of inducing points that equals the amount of data, $q(f)$ contains the true posterior. That is, if $\boldsymbol{z} = \boldsymbol{x}$, it is possible to set $\boldsymbol{m}$ and $\mathbf{S}$ such that $q(f) = p(f \mid \boldsymbol{y})$ [Titsias, 2009]. Secondly, David R. Burt et al. [2019] showed that for $N \to \infty$, the approximate posterior can be made arbitrarly close to the true posterior for $M = \mathcal{O}(\log N)$. Finally, we notice that the approximate mean exhibits the same structure as a parametric model with basis functions $\mathbf{k}_{\boldsymbol{u}\cdot}$ and weights $\mathbf{K}_{\boldsymbol{uu}}^{-1}\boldsymbol{m}$. The variance, however, remains non-parametric, which means that predictions are made with an infinite amount of basis functions—exactly like in the true posterior. For non-degenarate kernels, this leads to error bars that are unconstrained by the data. We will come back to the parameteric nature of the approximate mean in chapter 4.

We optimise the variational parameters $\{\boldsymbol{m}, \mathbf{S}\}$ by maximising the ELBO eq. (2.10). Assuming a general likelihood of the form $p(\boldsymbol{y} \mid f) = \prod_i p(y_i \mid f(x_i))$ the objective can be written as

$$\mathrm{ELBO} = \sum_{i=1}^{N} \mathbb{E}_{q(f(x_i))}[\log p(y_i \mid f(x_i))] - \mathrm{KL}[q(\boldsymbol{u}) \,\|\, p(\boldsymbol{u})]\,. \tag{2.14}$$

Crucially, the original KL between the two infinite-dimensional processes $\mathrm{KL}[q(f) \,\|\, p(f)]$ is mathematically equivalent to the finite-dimensional KL between the variational posterior and prior over the inducing variables. We refer the interested reader to Matthews et al. [2016] for a theoretical analysis of the equivalence and to Leibfried et al. [2020, Section 4.1] for a more intuitive explanation. The objective in eq. (2.14), introduced in Hensman et al. [2013; 2015] reduces the computational complexity to $\mathcal{O}(M^2N + M^3)$. It also allows for stochastic estimation through minibatching [Hensman et al., 2013] and for non-Gaussian likelihoods through Gauss-Hermite quadrature of the one-dimensinoal expectation over $q(f(x_i))$ [Hensman et al., 2015].

2.2.1 Interdomain Inducing Variables

Interdomain Gaussian processes use alternative forms of inducing variables such that the resulting sparse GP models consists of a different set of features. Classically, inducing variables are defined as pseudo function evaluations: $u_m = f(z_m)$, but in interdomain GPs the inducing variables are obtained using a linear transformation of the GP: $u_m = \mathcal{L}_m f$. This redefinition of $\boldsymbol{u}$ implies that the expressions of $\mathbf{K}_{\boldsymbol{uu}}$ and $\mathbf{k}_{\boldsymbol{u}\cdot}$ change, but the inference scheme of interdomain GPs and the mathematical expressions for the posterior mean and variance are exactly the same as classic sparse GPs. This is thanks to the linearity of the operator and the preservation of (joint) Gaussian behaviour under linear transformations. A common linear operator is the integral operator with an inducing function g_m [Lázaro-Gredilla and Figueiras-Vidal, 2009]:

$$\mathcal{L}_m f = \int f(x)\, g_m(x)\, \mathrm{d}x\,.$$

In this case, $\mathbf{K}_{\boldsymbol{uu}}$ and $\mathbf{k}_{\boldsymbol{u}\cdot}$ take the form

$$[\mathbf{K}_{\boldsymbol{uu}}]_{m,m'} = \mathrm{Cov}(\mathcal{L}_m f, \mathcal{L}_{m'} f) = \int\int k(x,x') g_m(x) g_{m'}(x') \mathrm{d}x\, \mathrm{d}x' \tag{2.15}$$

and

$$[\mathbf{k}_{\boldsymbol{u}\cdot}]_m = \mathrm{Cov}(\mathcal{L}_m f, f)) = \int k(\cdot, x') g_{m'}(x') \mathrm{d}x'. \tag{2.16}$$

Most current interdomain methods are designed to improve computational properties [Hensman et al., 2018; David R Burt et al., 2020; Dutordoir et al., 2020a]. For example, Variational Fourier Features (VFF) [Hensman et al., 2018] is an interdomain method where the inducing variables are given by a Matérn RKHS inner product between the GP and elements of the Fourier basis:

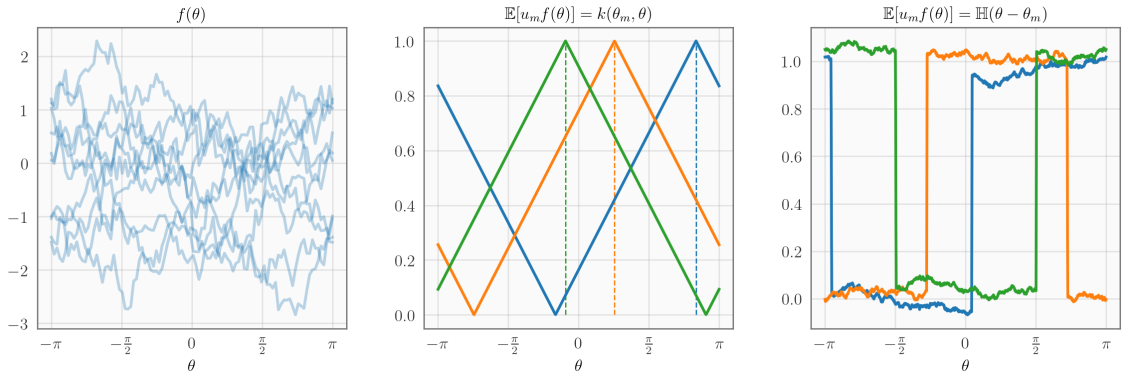
$$u_m = \langle f, \psi_m \rangle_{\mathcal{H}},$$

where $\psi_0 = 1$, $\psi_{2m} = \cos(mx)$ and $\psi_{2m+1} = \sin(mx)$ if the input space is $[0, 2\pi]$. This leads to

$$\mathbf{K}_{\boldsymbol{uu}} = [\langle \psi_i, \psi_j \rangle_{\mathcal{H}}]_{i,j=0}^{M-1} \qquad \text{and} \qquad \mathbf{k}_{\boldsymbol{u}\cdot} = [\psi_i(x)]_{i=0}^{M-1}\,.$$

This results in several advantages. First, the features $\mathbf{k}_{\boldsymbol{u}}$. are exactly the elements of the Fourier basis, which are independent of the kernel parameters and can be precomputed. Second, the matrix $\mathbf{K}_{\boldsymbol{uu}}$ is the sum of a diagonal matrix plus low rank matrices. This structure can be used to drastically reduce the computational complexity, and the experiments showed one or two orders of magnitude speed-ups compared to classic sparse GPs. Another reason to use interdomain inducing variables is the ability is gives to control $\mathbf{k}_{\boldsymbol{u}}$.—as shown in the next example.

Example: Heavyside Inducing Variable



(a) Samples (b) $u_m = f(\theta_m)$ (c) $u_m = \mathcal{L}_m f$

Fig. 2.1 Example of interdomain inducing variables for the first-order Arc Cosine kernel.

Using interdomain inducing variables in Sparse Variational Gaussian processes makes it possible to construct approximate GPs in which the basis functions exhibit interesting behaviour. As an illustration, in this example we design an inducing variable, through a linear transformation of the GP, for which the basis functions $\mathbf{k}_{\boldsymbol{u}}$. are given by the Heavide function. Therefore, assume $f \sim \mathcal{GP}(0, k)$ defined on $\mathbb{S}^1$ (the unit circle), $f : [-\pi, \pi] \to \mathbb{R}, \theta \mapsto f(\theta)$ and k the zero-th order Arc Cosine kernel [Cho and Saul, 2009], defined as:

$$k(\theta, \theta') = 1 - \frac{\varphi}{\pi}, \tag{2.17}$$

where $\varphi \in [0, \pi]$ is the shortest angle between the two inputs. In fig. 2.1a we show samples from this GP and in fig. 2.1b we plot $k(\theta_m, \theta)$ for three different values of θ_m.

Let us now define the m-th linear operator as the sum of the derivate at θ_m and the integral over the domain:

$$\mathcal{L}_m = \frac{\mathrm{d}}{\mathrm{d}\theta}\Big|_{\theta=\theta_m} + \frac{\pi}{2}\int_{-\pi}^{\pi} \mathrm{d}\theta, \tag{2.18}$$

and $u_m = \mathcal{L}_m f$, then the covariance between f and u_m is given by

$$
\begin{aligned}
[\mathbf{k}_{\boldsymbol{u}\cdot}]_m = \mathrm{Cov}(u_m, f(\cdot)) &= \mathbb{E}_f[\mathcal{L}_m f\, f(\cdot)] && \text{Definition covariance} && (2.19)\\
&= \mathcal{L}_m k(\theta, \cdot) && \text{Swap order } \mathcal{L}_m \text{ and } \mathbb{E}_f && (2.20)\\
&= \frac{\mathrm{d}}{\mathrm{d}\theta}\Big|_{\theta=\theta_m} k(\theta, \cdot) + \frac{\pi}{2}\int_{-\pi}^{\pi} k(\theta, \cdot)\mathrm{d}\theta && \text{Definition } \mathcal{L}_m && (2.21)\\
&= \mathbb{H}(\cdot - \theta_m) && \text{Computation.} && (2.22)
\end{aligned}
$$

The last step follows from the fact that derivative of k w.r.t. θ is -1 for $\theta < \theta_m$ and $+1$ otherwise. The integral part of the $\mathcal{L}_m$ is constant and simply shifts the covariance by $+1$, such that $\mathbf{k}_{\boldsymbol{u}\cdot}$ equals the Heaviside function. In fig. 2.1c, we show $[\mathbf{k}_{\boldsymbol{u}\cdot}]_m = \mathrm{Cov}(u_m, f(\theta))$ computed emperically using 1000 samples from the GP for three different values of θ_m. Using this construction and noting the parametric form of the mean of $q(f)$ (see eq. (2.13)) we created an approximate posterior GP that is equivalent to a linear combination of Heaviside functions. In chapter 4 we will employ a similar approach to construct a GP where the basis functions are ReLU.

2.3 Kernels and Reproducing Kernel Hilbert Space

Kernel have a rich history in machine learning and functional analysis, and have enabled the developement of versatile algorithms to analyse and process data. Kernels became widely noticed by the ML community through Support Vector Machines (SVMs), who prior to the deep learning revolution, dominated many machine learning benchmarks. For instance, the winner of the 2011 ImageNet Large Scale Visual Recognition Challenge (ILSVRC) was an SVM that used a Fisher kernel [Perronnin et al., 2010]. The year after, however, the same competition was—by a large margin—won by a deep convolutional neural network approach [Krizhevsky et al., 2012, AlexNet]. Nonetheless, kernels have always remained relevant and have surfaced in many other methods in the literature, such as splines, Priniciple Component Analysis (PCA) and Gaussian processes. Today, kernels are seen as an important tool to understand and analyse the behaviour of (deep) neural networks [e.g., Jacot et al., 2018]. In what follows, we are going to focus on the theoretical properties of Mercer kernels with the application of Gaussian processes in mind. Some of the theorems and definitions in this section are quoted from the excellent course "Machine learning with kernel methods"[1] by Julien Mairal and Jean-Philippe Vert, which I voluntarily auditted during my first year. Another excellent resource for this topic is Kanagawa et al. [2018].

Definition 1. *A positive definite (p.d.) kernel (or Mercer kernel) on a set $\mathcal{X}$ is a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that is*

[1] https://members.cbio.mines-paristech.fr/~jvert/svn/kernelcourse/course/2021mva

1. *symmetric* $k(x, x') = k(x', x)$*, and*
2. *satisfies for all* $x_i \in \mathcal{X}$ *and* $c_i \in \mathbb{R}$*:* $\sum_i \sum_j c_i c_j k(x_i, x_j) \geq 0$*.*

One of the simplest examples of a valid p.d. kernel is the linear kernel $k(x, x') = xx'$ with $\mathcal{X} = \mathbb{R}$. In the next theorem we will see that kernels induce a space over functions. For the linear kernel, for example, this space will consist of linear functions of the form $x \mapsto ax + b$. For more complicated kernels, however, it will be harder to explicity formulate the space of functions they induce.

Theorem 1 (Aronszajn [1950])**.** *The kernel* k *is a p.d. kernel on* $\mathcal{X}$ *i.f.f. there exists a Hilbert space* $\mathcal{H}$ *and a mapping* $\phi : \mathcal{X} \to \mathcal{H}$ *such that* $\forall x, x' \in \mathcal{X}$*:*

$$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}} \tag{2.23}$$

Intuitively, a kernel corresponds to the inner product after embedding the data in some Hilbert space. We want to remind the reader that a Hilbert space $\mathcal{H}$ is a (possibly infinite) vector space with an inner product $\langle f, g \rangle_{\mathcal{H}}$ and norm $||f|| = \langle f, f \rangle_{\mathcal{H}}^{\frac{1}{2}}$, and where any Cauchy sequence in $\mathcal{H}$ converges in $\mathcal{H}$. A Cauchy sequence is a sequence whose elements become arbitrarly close as the sequence progresses. A Hilbert space can be seen as a generalisation of an Euclidian space that is infinite dimensional. In the machine learning literature, $\mathcal{H}$ is also commonly referred to as the feature space and ϕ the feature map.

A Reproducing Kernel Hilbert Space (RKHS) is a special type of Hilbert space mentioned in theorem 1. It is a space over function from $\mathcal{X}$ to $\mathbb{R}$ where certain elements are "parameterised" by an element in $\mathcal{X}$. In other words, a datapoint $x \in \mathcal{X}$ is mapped to a function in the RKHS: $\phi(x) \in \mathcal{H}$. To make this more explicit we write $\phi(x) = k_x(\cdot)$ to highlight that $k_x(\cdot)$ is another function in $\mathcal{H}$.

Definition 2. *Let* $\mathcal{X}$ *be a set and* $\mathcal{H}$ *a Hilbert space with inner-product* $\langle \cdot, \cdot \rangle_{\mathcal{H}}$*. The function* $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *is called a reproducing kernel of* $\mathcal{H}$ *if*

1. $\mathcal{H}$ *contains all functions of the form*

$$\forall x \in \mathcal{X}, \quad k_x : \mathcal{X} \to \mathbb{R}, x' \mapsto k(x, x') \tag{2.24}$$

2. *For every* $x \in \mathcal{X}$ *and* $f \in \mathcal{H}$ *the reproducing property holds:*

$$f(x) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}} \tag{2.25}$$

If a reproducing kernel exists, then $\mathcal{H}$ *is called a reproducing kernel Hilbert space (RKHS).*

As a result of the reproducing property, remark that k can be written as an inner product in $\mathcal{H}$:

$$k(x, x') = \langle k(x, \cdot), k(x', \cdot) \rangle_{\mathcal{H}}, \quad \text{for } x, x' \in \mathcal{X}, \tag{2.26}$$

and therefore $k(x, \cdot)$ is sometimes referred to as the canonical feature map of x.

By the Moore-Aronszajn theorem [Aronszajn, 1950], it can be shown that the reproducing kernel belonging to an RKHS is unique and, conversely, that a function can only be the reproducing kernel of a single RKHS. Furthermore, a function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a positive definite kernel if and only if it is a reproducing kernel. In other words, there exists a one-to-one mapping between p.d. kernels and *their* RKHSs.

We can define the RKHS, associated to a p.d. kernel k, as follows:

$$\mathcal{H} = \left\{ f = \sum_{i=1}^{\infty} c_i k(x_i, \cdot), \forall c_i \in \mathbb{R}, x_i \in \mathcal{X} : \quad ||f||_{\mathcal{H}}^2 = \sum_{i,j} c_i c_j k(x_i, x_j) < \infty \right\} \tag{2.27}$$

where the inner product between $f = \sum_i a_i k(x_i, \cdot)$ and $g = \sum_j b_j k(x_j, \cdot)$ is given by $\langle f, g \rangle_{\mathcal{H}} = \sum_{i,j} a_i b_j k(x_i, x_j)$ and the norm is induced by the inner product $||f||_{\mathcal{H}}^2 = \langle f, f \rangle_{\mathcal{H}}$.

2.3.1 Spectral Formulation

Definition 3 (Kernel Operator)**.** *Associated to a kernel k we can define the kernel operator*

$$\mathcal{K} f = \int k(\cdot, x) f(x) \mathrm{d}\nu(x), \tag{2.28}$$

where ν denotes a measure. In our case this will typically be the Lebesgue measure or depend on the input distribution: $\nu(x) = p(x)\mathrm{d}x$.

It can be shown that for p.d. kernels the associated operator $\mathcal{K}$ is compact, positive (i.e. $\langle f, \mathcal{K} f \rangle_{\mathcal{H}} \geq 0$) and self-adjoint (i.e. $\langle f, \mathcal{K} g \rangle_{\mathcal{H}} = \langle \mathcal{K} f, g \rangle_{\mathcal{H}}$). Then according to the spectral theorem [e.g., Lang, 1993, Chapter 17] the operator can be diagonalised as there exists a complete orthonormal set $\{\phi_1, \phi_2, \ldots\}$ of eigenfunctions in $\mathcal{H}$ with real and non-negative eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$. This corresponds to

$$\mathcal{K} \phi_n = \lambda_n \phi_n \quad \text{and} \quad \langle \phi_n, \phi_m \rangle_{\mathcal{H}} = \delta_{nm}, \tag{2.29}$$

where $\delta_{nm} = 1$ if $n = m$ and 0 otherwise. The number of eigenfunctions and associated eigenvalues can be countably infinite $\mathbb{N}$ or finite $\{1, \ldots, N\}$. This result is analogues to the fact that semi-definite matrices can be decomposed into a finite set of eigenvectors and non-negative eigenvalues and where the eigenvectors form a basis for the space.

Mercer's theorem describes the kernel k in terms of $\{(\lambda_n, \phi_n)\}$, and as such opens interesting connections to GPs.

Theorem 2 (Mercer)**.** *Let $\mathcal{X}$ be a compact metric space, ν a positive measure on $\mathcal{X}$ and $\{(\lambda_n, \phi_n)\}$ the eigensystem of a p.d. kernel as described above, then*

$$k(x, x') = \sum_n \lambda_n \phi_n(x) \phi_n(x'), \quad \textit{for } x, x \in \mathcal{X} \tag{2.30}$$

where the convergence is absolute and uniform.

It is important to note that $\{(\lambda_n, \phi_n)\}$ will depend on the measure $\nu(x)$, which implicitly also depends on the domain $\mathcal{X}$. Furthermore, while Mercer's theorem is a very powerful result, there are only a few cases in which it is possible to compute the eigensystem associated with a kernel. In chapter 3 we discuss such as case.

We can use the eigen-decomposition of the kernel operator to represent the RKHS of a kernel

Theorem 3 (Mercer Representation of an RKHS)**.** *Let $\mathcal{X}$ be a compact metric space, k a p.d. kernel and $\{(\lambda_n, \phi_n)\}$ the eigensystem of the kernel operator for positive measure ν, then the RKHS of k is given by*

$$\mathcal{H} = \left\{ f = \sum_n a_n \phi_n : ||f||_{\mathcal{H}} = \sum_n \frac{a_n^2}{\lambda_n} < \infty \right\}, \tag{2.31}$$

with an inner product between $f, g \in \mathcal{H}$ given by

$$\langle f, g \rangle_{\mathcal{H}} = \frac{a_n b_n}{\lambda_n}, \quad \textit{where } f = \sum_n a_n \phi_n \textit{ and } g = \sum_n b_n \phi_n. \tag{2.32}$$

The reproducing property of the Mercer representation of an RKHS is straightforward to show. Let $f \in \mathcal{H}$ have coefficients $\{a_n\}$ and k be the kernel associated to $\mathcal{H}$ of which we assume that the Mercer representation is known, then

$$\langle f, k(x, \cdot) \rangle_{\mathcal{H}} = \sum_n \frac{a_n \lambda_n \phi_n(x)}{\lambda_n} = \sum_n a_n \phi_n(x) = f(x). \tag{2.33}$$

Through the Mercer decomposition of a kernel we can define a Gaussian process, this is known as the Karhunen-Loève expansion. We can define $f \sim \mathcal{GP}(0, k)$ as

$$f = \sum_n \sqrt{\lambda_n} \xi_n \phi_n \quad \text{with} \quad \xi_n \sim \mathcal{N}(0, 1) \tag{2.34}$$

as indeed the covariance between two points is given by

$$\text{Cov}(f(x), f(x')) = \sum_{n,m} \sqrt{\lambda_m} \sqrt{\lambda_n} \mathbb{E}[\xi_n \xi_m] \phi_n(x) \phi_m(x') = \sum_n \lambda_n \phi_n(x) \phi_n(x') = k(x, x'), \tag{2.35}$$

as a consequence of the i.d.d. and unit variance of the random variables $\{\xi_n\}$.

Using the Karhunen-Loève expression to compute the RKHS norm of a GP sample f yields $||f||_{\mathcal{H}}^2 = \sum_n \xi_n^2$, which is a diverging series. This shows that GP samples do not belong to the RKHS of their defining kernel, which may seem surprising. A straightforward solution to this problem is to truncate the Karhunen-Loève expansion and to only use the first $\tilde{N}$ eigenvalues and functions: $f = \sum_n^{\tilde{N}} \sqrt{\lambda_n} \xi_n \phi_n$, which approximates the kernel and the corresonponding GP.

Chapter 3

Linear Time Gaussian Processes on Spheres

Stationary kernels, i.e. translation invariant covariances, are ubiquitous in machine learning when the input space is Euclidean. When working on the hypersphere, their spherical counterpart are dot-product kernels, which are invariant to rotations. They are the main object under study in this chapter. In section 3.1, we first show how we can construct the Reproducing Kernel Hilbert Space (RKHS) of dot-product kernel on the hypersphere. Section 3.2 uses the RKHS to construct an efficient variational interdomain inducing variable for Gaussian processes on the hypersphere. We will then show how to expand the GP onto the complete domain $\mathbb{R}^d$ and conclude with a series of experiments, showing the speed and accuracy of the proposed approach.

3.1 Mercer Representation of Dot-Product Kernels

Consider the unit sphere in $\mathbb{R}^d$ as the input domain

$$\mathcal{X} = \mathbb{S}^{d-1} = \{x \in \mathbb{R}^d : ||x||_2 = 1\} \tag{3.1}$$

and ν to be the Lebesgue measure on $\mathbb{S}^{d-1}$, so that

$$\Omega_{d-1} = \int_{\mathbb{S}^{d-1}} \mathrm{d}\nu(x) = \frac{2\pi^{d/2}}{\Gamma(d/2)}. \tag{3.2}$$

is the surface area of $\mathbb{S}^{d-1}$. We adopt the usual L_2 inner product for functions $f : \mathbb{S}^{d-1} \to \mathbb{R}$ and $g : \mathbb{S}^{d-1} \to \mathbb{R}$ restricted to the sphere

$$\langle f, g \rangle_{L_2(\mathbb{S}^{d-1})} = \frac{1}{\Omega_{d-1}} \int_{\mathbb{S}^{d-1}} f(x)\, g(x)\, \mathrm{d}\nu(x). \tag{3.3}$$

We define a dot-product kernel, also known as a zonal kernel (we will use both terms interchangably), as a p.d. kernel of the form

$$k(x, x') = \kappa(x^\top x'), \tag{3.4}$$

where $\kappa : [-1, 1] \to \mathbb{R}$ is a continuous function and referred to as the shape function. In other words, dot-product kernels only depend on the distance on the great circle between two inputs, rather than their location. They are the counterpart of stationary kernels $k(x, x') = \kappa(x - x')$, who are functions of the difference only. For example, where stationary kernels are translation invariant, dot-product kernel are rotationally invariant.

We can associate a kernel operator $\mathcal{K}$ to a zonal kernel, as explained in section 2.3.1, which exhibits the form

$$\mathcal{K}f = \int_{\mathbb{S}^{d-1}} \kappa(x^\top \cdot) f(x) \mathrm{d}\nu(x). \tag{3.5}$$

To construct a Mercer representation of a zonal kernel's RKHS we require the eigensystem of the kernel operator $\mathcal{K}$, or, equivalently, the set of eigenfunctions $\{\phi_n\}$ and associated eigenvalues $\{\lambda_n\}$ for which

$$\mathcal{K}\phi_n = \lambda_n \phi_n \qquad \text{and} \qquad \langle \phi_n, \phi_m \rangle_{L_2(\mathbb{S}^{d-1})} = \delta_{nm}. \tag{3.6}$$

To obtain the eigensystem of $\mathcal{K}$ we will show that it commutes with the Laplace-Beltrami operator $\Delta^{\mathbb{S}^{d-1}}$, henceforth referred to as simply Laplacian or Laplace operator. We want to remind the reader that commuting operators share the same eigenfunctions, but not necessarily the same eigenvalues. Therefore, to find the eigenfunctions of the kernel operator $\mathcal{K}$, it suffice to find the eigenfunctions of the Laplacian. Fortunately, there exists a huge body of literature on the diagonalisation of the Laplace operator on $\mathbb{S}^{d-1}$, and it is well-known that its eigenfunctions are given by the spherical harmonics. In the following paragraph we first proof the commutativity of the operators before coving RKHS and the spherical harmonic basis functions.

It now rest us to show that the Laplacian and the kernel operator commute. For this, we first show that for zonal kernels and $x, s \in \mathbb{S}^{d-1}$ the following property holds

$$\begin{aligned}
\Delta_x^{\mathbb{S}^{d-1}} k(x, s) &= \nabla_x \cdot \nabla_x \kappa(x^\top s) && \text{Definition } \Delta^{\mathbb{S}^{d-1}} \text{ and zonal kernels} && (3.7)\\
&= \nabla_x \cdot (s\kappa'(x^\top s)) && \text{Chainrule} && (3.8)\\
&= ||s||^2 \kappa''(x^\top s) = \kappa''(x^\top s) && \text{Because } ||s|| = 1. && (3.9)
\end{aligned}$$

Similarly, for $\Delta_s^{\mathbb{S}^{d-1}} k(x, s)$ we obtain $k''(x^\top s)$, and as a result

$$\Delta_x^{\mathbb{S}^{d-1}} k(x, s) = \Delta_s^{\mathbb{S}^{d-1}} k(x, s). \tag{3.10}$$

We now show that the Laplacian and the kernel operator of zonal kernels commute

$$\mathcal{K}\left[\Delta^{\mathbb{S}^{d-1}} f\right] = \int_{\mathbb{S}^{d-1}} k(x,s)\left[\Delta_x^{\mathbb{S}^{d-1}} f(x)\right] \mathrm{d}\nu(x) \qquad \text{Definition } \mathcal{K}, \text{ eq. (3.5)} \qquad (3.11)$$

$$= \int_{\mathbb{S}^{d-1}} f(x)\Delta_x^{\mathbb{S}^{d-1}} k(x,s)\mathrm{d}\nu(x) \qquad \text{Integration by parts} \qquad (3.12)$$

$$= \int_{\mathbb{S}^{d-1}} f(x)\Delta_s^{\mathbb{S}^{d-1}} k(x,s)\mathrm{d}\nu(x) = \Delta^{\mathbb{S}^{d-1}}\mathcal{K}f \qquad \text{eq. (3.10)}. \qquad (3.13)$$

which in turn implies that these two operators share the same eigenfunctions. This result is of particular relevance to us since there is a huge body of literature on diagonalisation of the Laplace-Beltrami operator on $\mathbb{S}^{d-1}$, and that it is well known that its eigenfunctions are given by the spherical harmonics. This reasoning can be summarised by the following theorem:

Theorem 4 (Mercer representation)**.** *Any zonal kernel k on the hypersphere can be decomposed as*

$$k(x,x') = \sum_{\ell=0}^{\infty} \sum_{k=1}^{N_n^d} \widehat{a}_{\ell,k}\phi_{\ell,k}(\boldsymbol{x})\phi_{\ell,k}(\boldsymbol{x}'), \qquad (3.14)$$

where $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{S}^{d-1}$ and $\widehat{a}_{\ell,k}$ are positive coefficients, $\phi_{\ell,k}$ denote the elements of the spherical harmonic basis in $\mathbb{S}^{d-1}$, and N_ℓ^d corresponds to the number of spherical harmonics for a given level ℓ.

Although it is typically stated without a proof, this theorem is already known in some communities (see Wendland [2005] for a functional analysis exposition, or Peacock [1999] for its use in cosmology).

1. what is a dot-product kernel
2. example 1: (deep) arc cosine
3. example 2: stationary kernels
4. decomposition
5. proof
6. refer to spherical harmonics

3.2 Variational Spherical Harmonic Gaussian Processes

3.2.1 Homogeneous Extension to $\mathbb{R}^d$

3.3 Experiments

3.4 Spherical Harmonics

Spherical harmonics are a special set of functions defined on the hypersphere and play a central role in harmonic analysis and approximation theory [Wendland, 2005]. They originate from solving Laplace's equation, and form a complete set of orthogonal functions. Any sufficiently regular function defined on the sphere can be written as a sum of these spherical harmonics, similar to the Fourier series with sines and cosines. Spherical harmonics are defined in arbitrary dimensions [Efthimiou and Frye, 2014; Dai and Xu, 2013], but lack explicit formulations and practical implementations in dimensions larger than three.

In this section, we propose a novel algorithm to construct spherical harmonics in d dimensions. The algorithm is based on the existence of a fundamental system of points on the hypersphere, which we select in a greedy fashion through optimisation. This result in spherical harmonics that are a linear combination of zonal functions and form an orthnormal basis on $\mathbb{S}^{d-1}$. The algorithm lends itself well for implementation in Python and TensorFlow, which we provide at: https://github.com/vdutor/SphericalHarmonics. The code is accompagnied by a series of tests that show that the properties of spherical harmonics, as detailed below, hold. Before outlining the algorithm, we briefly define and cover the important properties of spherical harmonics in $\mathbb{R}^d$. We refer the interested reader to Dai and Xu [2013] and Efthimiou and Frye [2014] for a comprehensive overview.

We adopt the usual L_2 inner product for functions $f : \mathbb{S}^{d-1} \to \mathbb{R}$ and $g : \mathbb{S}^{d-1} \to \mathbb{R}$ restricted to the sphere

$$\langle f, g \rangle_{L_2(\mathbb{S}^{d-1})} = \frac{1}{\Omega_{d-1}} \int_{\mathbb{S}^{d-1}} f(x)\, g(x)\, \mathrm{d}\omega, \tag{3.15}$$

where $\mathrm{d}\omega(x)$ is the surface area measure such that Ω_{d-1} denotes the surface area of $\mathbb{S}^{d-1}$

$$\Omega_{d-1} = \int_{\mathbb{S}^{d-1}} \mathrm{d}\omega(x) = \frac{2\pi^{d/2}}{\Gamma(d/2)}. \tag{3.16}$$

Throughout this section we use the following notation and definitions. For $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ and $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}^d$, a monomial x^α is a product $x^\alpha = x_1^{\alpha_1} \ldots x_d^{\alpha_d}$, which has degree $|\alpha| = \alpha_1 + \ldots \alpha_d$. A real homogeneous polynomial $P(x)$ of degree n is a linear combination of monomials of degree n with real coefficients, that is $P(x) = \sum_{|\alpha|=n} c_\alpha x^\alpha$, with $c_\alpha \in \mathbb{R}$. We denote $\mathcal{P}_n^d$ as the space of real homogeneous polynomials of degree n, and can show that, counting the cardinality of the set $\{\alpha \in \mathbb{N}^d : |\alpha| = n\}$, that $\dim(\mathcal{P}_n^d) = \binom{n+d-1}{n}$. A function

$f : \mathbb{R}^d \to \mathbb{R}$ is said to be *harmonic* if $\Delta f = 0$, where $\Delta = \partial_{x_1}^2 + \ldots + \partial_{x_d}^2$ and ∂_{x_i} the partial derivate w.r.t. the i-th variable.

Definition 4. *The spherical harmonics of degree n of d variables, denoted by $\mathcal{H}_n^d$, is the linear space of harmonic and homogeneous in degree n polynomials on $\mathbb{S}^{d-1}$, that is*

$$\mathcal{H}_n^d = \{p \in \mathcal{P}_n^d : \Delta p = 0 \text{ and } p : \mathbb{S}^{d-1} \to \mathbb{R}\}. \tag{3.17}$$

The dimensionality of $\mathcal{H}_n^d$ is given by

$$dim(\mathcal{H}_n^d) = \frac{2n+d-2}{n}\binom{n+d-3}{d-1} := N_n^d. \tag{3.18}$$

The space $\mathcal{H}_n^d$ has an orthonormal basis consisting of N_n^d functions, denoted by $\{\phi_{n,j}\}_{j=1}^{N_n^d}$. The basis satisfy the following properties

$$\mathcal{H}_n^d = \text{span}\Big(\phi_{n,1}, \ldots, \phi_{n,N_n^d}\Big), \quad \text{and} \quad \langle \phi_{n,j}, \phi_{n',j'} \rangle_{L_2(\mathbb{S}^{d-1})} = \delta_{nn'}\delta_{jj'}. \tag{3.19}$$

From the completeness and orthonormality of the spherical harmonic basis $\{\phi_{n,j}\}_{n=0,j=1}^{\infty,N_n^d}$, it can be shown that they also form a basis of square-integrable functions [Efthimiou and Frye, 2014]. This means that we can decompose a function $f : \mathbb{S}^{d-1} \to \mathbb{R}$ as

$$f = \sum_{n=0}^{\infty}\sum_{j=1}^{N_n^d} \widehat{f}_{n,j}\phi_{n,j}, \quad \text{with} \quad \widehat{f}_{n,j} = \langle f, \phi_{n,j} \rangle_{L_2(\mathbb{S}^{d-1})}, \tag{3.20}$$

which can be seen as the spherical analogue of the Fourier decomposition of periodic functions onto a basis of sines and cosines.

Subsequently, we will coin the set $\{\phi_{n,j}\}$ as the spherical harmonics. They are indexed by n and j, where $n = 0, 1, 2, \ldots$ denotes the degree (or level) and $j = 1, \cdots, N_n^d$ denotes the orientation of the spherical harmonic. We are interested in finding $\{\phi_{n,j}\}$ in arbitrary dimension. For $d = 2$, is solving Laplace's equation ($\Delta p = 0$) directly relatively straightforward. Doing so reveals that $N_0^2 = 1$ with $\phi_{0,1} = 1$ and $N_n^2 = 2$ for all $n > 0$ with $\phi_{n,1}(\theta) = \sqrt{2}\cos(n\theta)$ and $\phi_{n,2}(\theta) = \sqrt{2}\sin(n\theta)$. This shows that on the unit circle $\mathbb{S}^1$, the spherical harmonics correspond to the Fourier basis. For $d = 3$, we can also directly solve Laplace's diffential equation to find $N_n^d = 2n + 1$ and a closed form solution for $\{\phi_{n,j}\}$. However, for $d > 3$, explicit formulations for the spherical harmonics become very rare. To the best of our knowledge, the only explicit formulation we could find is in Dai and Xu [2013, Theorem 5.1], which consists of a product over polynomials. This makes the implementation cumbersome and numerically unstable, and only practically useful up to 10 dimensions [Dutordoir et al., 2020a]. However, making use of the following two theorems, in **??** we can derive another formulation for the basis of spherical

harmonics as a sum of polynomials, rather than a product. The connection between spherical harmonics and orthogonal polynomials becomes clear in the next theorem.

Theorem 5 (Addition)**.** *Let $\{\phi_{n,j}\}_{j=1}^{N_n^d}$ be an orthonormal basis for the spherical harmonics of degree n and $x, x' \in \mathbb{S}^{d-1}$. Then the Gegenbauer polynomial $C_n^{(\alpha)} : [-1,1] \to \mathbb{R}$ of degree n can be written as*

$$\sum_{j=1}^{N_n^d} \phi_{n,j}(x)\phi_{n,j}(x') = \frac{n+\alpha}{\alpha} C_n^{(\alpha)}(x^\top x') \quad \textit{with} \quad \alpha = \frac{d-2}{2}. \tag{3.21}$$

As a result of the relation between the Gegenbauer polynomial and the spherical harmonics, are the Gegenbauer polynomial sometimes referred to as ultraspherical polynomials. For $d = 2$, Theorem 1 recovers the addition formula of the cosine function, as indeed $\cos(\theta)\cos(\theta') + \sin(\theta)\sin(\theta') = \cos(\theta - \theta')$ and $C_n^{(0)}(t) = \cos(n \arccos(t))$. The Gegenbauer polynomials with $\alpha = 0$ are better known as the Chebyshev polynomials. Another connection between spherical harmonics and Gegenbauer polynomials is given by the Funk-Hecke theorem and applies to zonal functions. A zonal function on $\mathbb{S}^{d-1}$ is a function that is rotationally invariant w.r.t. to a point on the sphere, $\eta \in \mathbb{S}^{d-1}$. This means that the function only depends on the inner product $\eta^\top x$, or equivalently, on the geodestic distance between η and x.

Theorem 6 (Funk-Hecke)**.** *Let f be an integrable function such that $\int_{-1}^{1} \|f(t)\|(1-t^2)^{(d-3)/2}\mathrm{d}t$ is finite and $d \geq 2$. Then for every $\phi_{n,j}$ and $\eta \in \mathbb{S}^{d-1}$*

$$\frac{1}{\Omega_{d-1}} \int_{\mathbb{S}^{d-1}} f(\eta^\top x)\, \phi_{n,j}(x)\, \mathrm{d}\omega(x) = \lambda_n\, \phi_{n,j}(\eta), \tag{3.22}$$

where λ_n is a constant defined by

$$\lambda_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{-1}^{1} f(t)\, C_n^{(\alpha)}(t)\, (1-t^2)^{\frac{d-3}{2}} \mathrm{d}t, \tag{3.23}$$

with $\alpha = \frac{d-2}{2}$ and $\omega_d = \frac{\Omega_{d-2}}{\Omega_{d-1}}$.

3.4.1 Zonal Spherical Harmonics

From the Funk-Hecke and the Addition theorem, it is clear that there is a strong connection between spherical harmonics and Gegenbauer polynomials. The next theorem develops this connection further as it states that a basis for spherical harmonics can be written as zonal Gegenbauer polynomials.

Theorem 7. *If $\{\eta_1, \ldots, \eta_{N_n^d}\} \in \mathbb{S}^{d-1}$ is a fundamental system of points on the sphere, then $\{C_n^{(\alpha)}(\eta_i \cdot)\}_{i=1}^{N_n^d}$ is a basis for $\mathcal{H}_n^d$. A collection of points $\{\eta_1, \ldots, \eta_M\} \in \mathbb{S}^{d-1}$ is called a funda-*

mental system of degree n consisting of M points on the sphere if

$$det \begin{bmatrix} C_n^{(\alpha)}(1) & \dots & C_n^{(\alpha)}(\eta_1^\top \eta_M) \\ \vdots & & \vdots \\ C_n^{(\alpha)}(\eta_M^\top \eta_1) & \dots & C_n^{(\alpha)}(1) \end{bmatrix} > 0. \tag{3.24}$$

Finding a basis for $\mathcal{H}_n^d$ is thus equivalent to finding a set of N_n^d points that satisfy eq. (3.24). Crucially, Dai and Xu [2013, Lemma 3] show that there always exists a fundamental system of degree n and N_n^d points.

Following the theoreom, if we wish to construct $\{\phi_{n,j}\}_{n,j}$, an *orthnormal* basis for the spherical harmonics, we firstly need a fundamental system of points. Secondly, while $\{C_n^{(\alpha)}(\eta_i \cdot)\}_{i=1}^{N_n^d}$ forms a basis for $\mathcal{H}_n^d$, the basis is not orthonormal. We will thus have to apply a Gram-Schmidt process for orthonormalising the basis. We detail both steps in the next paragraphs.

Construction of a fundamental system of points We propose to build a fundamental system of points in a greedy fashion by iteratively adding a point on the sphere that maximises the determinant as given in eq. (3.24). Therefore, let $\boldsymbol{\eta} = \{\eta_1, \dots, \eta_M\}$ contain the M points that are already in the fundamental system and define the following block-matrix of size $(M+1) \times (M+1)$ as

$$\mathbf{M}(\boldsymbol{\eta}, \eta_*) = \left[\begin{array}{c|c} C_n^{\alpha}(\boldsymbol{\eta}\boldsymbol{\eta}^\top) \in \mathbb{R}^{M \times M} & C_n^{(\alpha)}(\boldsymbol{\eta}\eta_*^\top) \in \mathbb{R}^{M \times 1} \\ \hline C_n^{(\alpha)}(\boldsymbol{\eta}\eta_*^\top)^\top \in \mathbb{R}^{1 \times M} & C_n^{(\alpha)}(1) \in \mathbb{R} \end{array}\right], \tag{3.25}$$

where $C_n^{(\alpha)}(\boldsymbol{\eta}\boldsymbol{\eta}^\top)$ corresponds to elementwise evaluating the Gegenbauer polynomial $C_n^{(\alpha)} : [-1, 1] \to \mathbb{R}$ for each element of $\boldsymbol{\eta}\boldsymbol{\eta}^\top \in \mathbb{R}^{M \times M}$. A new point η is added to the fundamental system if it maximises the determinant

$$\eta = \underset{\eta_* \in \mathbb{S}^{d-1}}{\operatorname{argmax}} \ \det(\mathbf{M}(\boldsymbol{\eta}, \eta_*)), \tag{3.26}$$

in order to satisfy the condition in eq. (3.24). Computing the determinant can be done efficiently using Schur' complement. Furthermore, as $\boldsymbol{\eta}$ and $C_n^{(\alpha)}(1.0)$ are constants the optimisation problem boils down to

$$\eta = \underset{\eta_* \in \mathbb{R}^d}{\operatorname{argmin}} \ C_n^{(\alpha)}(\frac{\eta_*}{||\eta_*||}\boldsymbol{\eta}^\top)\left[C_n^{(\alpha)}(\boldsymbol{\eta}\boldsymbol{\eta}^\top)\right]^{-1} C_n^{(\alpha)}(\boldsymbol{\eta}\frac{\eta_*^\top}{||\eta_*||}). \tag{3.27}$$

The complete algorithm is given in algorithm 1.

Algorithm 1: Construction of fundamental system

Input: Degree n and dimension d

Result: Fundamental system $\boldsymbol{\eta} = \{\eta_0, \dots \eta_{N_n^d}\}$

$\eta_1 = (0, 0, \dots, 1)$ `// d-dimensional vector pointing north`

$\boldsymbol{\eta} = \{\eta_1\}$, $\alpha = \frac{d-2}{2}$, $i = 2$

while $i \leq N_n^d$ **do**

  $\eta = \text{argmax}_{\eta_* \in \mathbb{R}^d} \det(\mathbf{M}(\boldsymbol{\eta}, \frac{\eta_*}{||\eta_*||}))$ `// Using a local optimisation method (e.g., BFGS) and eq. (3.27)`

  Add η to $\boldsymbol{\eta}$

  i = i + 1

end

Orthonormalisation Proof $\langle C_n^{(\alpha)}(\eta_i^\top \cdot), C_n^{(\alpha)}(\eta_j^\top \cdot)\rangle_{L_2(\mathbb{S}^{d-1})} = C_n^{(\alpha)}(\eta_i^\top \eta_j)$ as a result of the Funk-Hecke theorem.

Theorem 8. *Let* $\boldsymbol{\eta} = \{\eta_1, \dots, \eta_{N_n^d}\}$ *be a fundamental system of degree* n *consisting of* N_n^d *points, and* $\mathbf{L}$ *the inverse cholesky factor of* $C_n^{(\alpha)}(\boldsymbol{\eta}\boldsymbol{\eta}^\top)$*. Then for* $j = 1, \dots, N_n^d$ *and*

$$\phi_{n,j}(x) = \sum_{i=1}^{N_n^d} \mathbf{L}_{j,i}\, C_n^{(\alpha)}(\eta_i^\top x) \tag{3.28}$$

is $\{\phi_{n,j}\}$ *an orthnormal basis for the spherical harmonics* $\mathcal{H}_n^d$*.*

Chapter 4

Deep Neural Networks as Point Estimates for Deep Gaussian Processes

MacKay [1992a; 1992b] noted very early that the Bayesian treatment of neural networks had large potential. To start, it can help to quantify estimates of the error bars on the network outputs, but it can also provide an objective that can be used for the comparison of alternative network architectures. The reality, however, was that Bayesian inference in neural networks was –and still is– challenging, and in practice one has to resort to either crude approximations (e.g., Laplace approximation [MacKay, 1998]) or lengthy computations (e.g., Markov Chain Monte Carlo [Neal, 1992]).

Building on the foundational work of MacKay, and driven by the amazing successes of large deterministic deep neural networks (DNNs), the literature has seen an upspring in the development of more scalable methods to perform approximate Bayesian inference in DNNs [Blundell et al., 2015; Kingma et al., 2015; Gal and Ghahramani, 2016]. However, it remains challenging to encode prior assumptions on functions through distributions on weights and the large number of parameters to be estimated makes it computationally prohibitive. Furthermore, the strong approximations used both during modelling and inference make it unclear to what extent these models approximate the true posterior distribution [Hron et al., 2018; Foong et al., 2020]. They also do not deliver on an important promise of Bayesian methods: an approximate marginal likelihood objective that can be used for automatic model selection and hyperparameter learning. A different approach may thus be necessary to unlock the Bayesian benefits in deep learning.

Neal [1995] showed that for infinitely wide single-layer BNNs the distribution over the non-linear functions are given by Gaussian processes. Williams [1998] and Cho and Saul [2009] extended this theory and derived the kernel corresponding to an infinite-width BNN

with Sigmoidal and ReLU activation function, respectively. The beauty of this connection is that performing Bayesian inference in the corresponding GP model can be done exactly and analytically – all in a single elegant framework [Rasmussen and Williams, 2006]. Since, various approximations to the exact GP framework have been developed to allow for non-Gaussian likelihoods [Kuss and Rasmussen, 2005; Hensman et al., 2013], large datasets [Hensman et al., 2015; Wang et al., 2019], and even neural network like structures such as convolutions [van der Wilk et al., 2017]. Crucially, the approximations to the marginal likelihood still enable the main Bayesian benefits: model uncertainty and learning model hyperparameters (e.g., [van der Wilk et al., 2018; Dutordoir et al., 2020b]).

More recently, Matthews et al. [2018] discovered the equivalence between *deep* (i.e. multi-layer) BNNs and GPs. This has led to the development of deep (fully connected and convolutional) neural network kernels for GPs (NN-GPs). Interestingly, the performance of the non-Bayesian neural nets significantly outperforms the corresponding GPs [Garriga-Alonso et al., 2018; Novak et al., 2019]. The discrepancy hints at the fact that these single-layer GPs, even when configured with very expressive DNN equivalent kernels, are missing a crucial component: the ability to learn feature representations from the data.

Deep Gaussian processes [Damianou and Lawrence, 2013, DGPs] are an interesting avenue to tackle these challenges. They are built by hierarchically stacking GPs on top of each other, which enables the learning of more powerful representations through compositional features. Moreover, their Bayesian approximations in function-space seem to be of higher quality than those of weight-space BNNs, e.g. as supported by the successful use of marginal likelihood estimates for hyperparameter learning [Damianou and Lawrence, 2013].

While promising, DGPs have struggled for relevance in applications due to the challenges and cost associated with Bayesian inference. Training DGPs is computationally expensive and requires very careful setting of the parameters. Furthermore, the hierarchical prior induced by naively stacking stationary kernels gives rise to pathological, "collapsed" samples Duvenaud et al. [2014]. Considerable progress for scalable inference in DGPs was made by Hensman and Lawrence [2014] and Salimbeni and Deisenroth [2017], who derived stochastic variational lower bounds. The formulation of these bounds closely resembles the computations of training a feed-forward neural network with regularisation terms, and has greatly inspired this work.

Building on Salimbeni and Deisenroth [2017] and to further improve the scalability of DGPs, Cutajar et al. [2017] used a Random Fourier Feature [Rahimi and Recht, 2008, RFF] approximation of the kernel. While successful, this approach introduces an approximation in both the prior and the posterior of the model. More recently, Rudner et al. [2020] proposed Fourier features of the Matérn RKHS, following Hensman et al. [2018, Variational Fourier Features (VFF)], to build inter-domain DGPs. Like for single layer VFF models, this approach can lead to faster training and improved performance, but is only computationally feasible for data of dimension one or two. In parallel with our work, Sun et al. [2021] explored the idea of using single-layer neural networks to parameterise inducing points in shallow GPs. Similar

to this paper, their method makes use of the spherical harmonic decomposition of a kernel. However, their works differs in the fact that they focus on shallow GPs, on bounded inducing functions (e.g., erf), and directly use the Nyström approximation to approximate the model's uncertainty estimates rather than the ELBO to learn the variational parameters.

The analysis of (Bayesian) neural networks has led to several probabilistic models: NN-GPs, GPs and DGPs. In this work, however, rather than focusing on the *prior* induced by these equivalent models, the emphasis lies on the connection between the DGP *posterior* and the DNN. This has received much less attention in the literature, though we argue it is a more interesting regime to study. The connection between GP piors and neural nets is established only in the infinite limit of the number of hidden units. The sparse posterior DGP, on the contrary, is built out of a finite set of basis functions and can thus immediately be connected to finite-width neural networks — in this paper we connect both models in their modus operandi.

We formulate a DGP configuration for which the approximate posterior mean has the same mathematical structure as a deep neural network with fully connected layers and non-linear activation functions. We can use this unification to train the DGP like a neural network which allow us to leverage all the great research in this area for DGP inference. Furthermore, this connection between DGPs posteriors and DNNs highlights the great potential of DGPs as a model for learning powerful representations from data while being fully Bayesian.

4.1 Method: Gaussian Process Layers with Activated Inducing Variables

The concepts introduced in the background section can now be used to summarise our approach: We consider DGP models with GP layers that have an Arc Cosine kernel and Variational Fourier Feature style inducing variables $u_m = \langle f(\cdot), g_m(\cdot) \rangle_{\mathcal{H}}$ [Hensman et al., 2018]. We then choose inducing functions $g_m(\cdot)$ that have the same shape as neural network activation functions (section 4.1.1). This yields basis functions for the SVGP model that correspond to activation functions (section 4.1.2), and thus to a model whose mean function can be interpreted as a classic feed forward neural network. section 4.1.3 covers the mathematical intricacies associated with the construction described above.

4.1.1 Activated Inducing Functions and their Spherical Harmonic Decomposition

The RKHS of the Arc Cosine kernel consists solely of functions that are equal to zero at the origin, i.e. $\forall f \in \mathcal{H} : f(\mathbf{0}) = 0$. To circumvent this problem we artificially increase the input space dimension by concatenating a constant to each input vector. In other words, the data space is embedded in a larger space with an offset such that it does not contain the origin anymore. This is analogous to the bias unit in multi-layer perceptron (MLP) layers in neural

networks. For convenience we will denote by $(d-1)$ the dimension of the original data space (i.e. the number of input variables), and by d the dimension of the extended space on which the Arc Cosine kernel is defined.

The inducing functions $g_m(\cdot)$ play an important role because they determine the shape of the SVGP's basis functions. Ideally they should be defined such that their restriction to the $(d-1)$-dimensional original data space matches classic activation functions, such as the ReLU or the Softplus, exactly. However, this results in an angular component for $g_m(\cdot)$ that is not necessarily zonal. Since this property will be important later on, we favour the following alternative definition that enforces zonality:

$$g_m : \mathbb{R}^d \to \mathbb{R}, \qquad \boldsymbol{x} \mapsto ||\boldsymbol{x}||\,||\boldsymbol{w}_m||\,\sigma\left(\frac{\boldsymbol{w}_m^\top \boldsymbol{x}}{||\boldsymbol{w}_m||\,||\boldsymbol{x}||}\right), \tag{4.1}$$

with $\boldsymbol{w}_m \in \mathbb{R}^d$ a parameter, and $\sigma : [-1, 1] \to \mathbb{R}$ the function that determines the value of $g_m(\cdot)$ on the unit hypersphere. In **??**, we show that choosing $\sigma(\cdot)$ to be a ReLU ($\sigma(t) = \max(0, t)$) or a Softplus ($\sigma(t) = \log(1 + \exp(3\,t))$) activation function leads to inducing functions that closely resemble the classic ReLU and Softplus on the data space. In the specific case of the ReLU it can actually be shown that the match is exact. In both cases, The parameter $\boldsymbol{w}_m \in \mathbb{R}^d$ determines the orientation and slope of the activation function—they play the same role as the pre-activation weights in a NN.

The zonality that we enforced in eq. (4.1) is particularly convenient when it comes to representing $g_m(\cdot)$ in the basis of the eigenfunctions of $\mathcal{H}$, which is required for computing inner products. It indeed allows us to make use of the Funk-Hecke theorem (see **??**) and to eventually obtain

$$\begin{aligned} g_m(\boldsymbol{x}) = ||\boldsymbol{x}||\,||\boldsymbol{w}_m|| \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} \sigma_n \phi_{n,j}\left(\frac{\boldsymbol{w}_m}{||\boldsymbol{w}_m||}\right) \phi_{n,j}\left(\frac{\boldsymbol{x}}{||\boldsymbol{x}||}\right), \\ \text{where} \quad \sigma_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{-1}^{1} \sigma(t)\, C_n^{(\alpha)}(t)\, (1-t^2)^{\frac{d-3}{2}} \mathrm{d}t. \end{aligned} \tag{4.2}$$

Analytical expressions for σ_n when $\sigma(t) = \max(0, t)$ are given in **??**.

4.1.2 Activated Interdomain Inducing Variables

We define our *activated* interdomain inducing variables as

$$u_m = \langle f(\cdot), g_m(\cdot) \rangle_{\mathcal{H}}. \tag{4.3}$$

Since the GP samples do not belong to the RKHS there are mathematical subtleties associated with such a definition, which are detailed in section 4.1.3. Assuming for now that they are indeed well defined, using these interdomain inducing variables as part of the SVGP framework

requires access to two quantities: (i) their pairwise covariance, and (ii) the covariance between the GP and the inducing variables. The pairwise covariance, which is needed to populate $\mathbf{C}$, is given by

$$\mathrm{Cov}(u_m, u_{m'}) = \langle g_m(\cdot), g_{m'}(\cdot)\rangle_{\mathcal{H}} = \sum_{\substack{n=0 \\ \lambda_n \neq 0}}^{\infty} \frac{\sigma^2}{\lambda_n} \frac{n+\alpha}{\alpha} C_n^{(\alpha)}\left(\frac{\boldsymbol{w}_m^\top \boldsymbol{w}_{m'}}{||\boldsymbol{w}_m||\,||\boldsymbol{w}_{m'}||}\right). \tag{4.4}$$

The above is obtained using the RKHS inner product from **??**, the Fourier coefficients from eq. (4.2) and the addition theorem for spherical harmonics from **??**. Secondly, the covariance between the GP and u_m, which determines $[c_u(\cdot)]_m$, is given by:

$$\mathrm{Cov}(u_m, f(\boldsymbol{x})) = \langle k(\boldsymbol{x}, \cdot), g_m(\cdot)\rangle_{\mathcal{H}} = g_m(\boldsymbol{x}) \tag{4.5}$$

as a result of the reproducing property of the RKHS. It becomes clear that this procedure gives rise to basis functions that are equal to our inducing functions. By construction, these inducing functions match neural network activation functions in the data plane, as shown in **??**. Using these inducing variables thus leads to an approximate posterior GP (eq. (2.13)) which has a mean that is equivalent to a fully-connected layer with a non-linear activation function (e.g. ReLU, Softplus, Swish).

4.1.3 Analysis of the interplay between kernels and inducing functions

In this section we describe the mathematical pitfalls that can be encountered [e.g., Sun et al., 2021] when defining new inducing variables of the form of eq. (4.3), and how we address them. We discuss two problems: 1) the GP and the inducing function are not part of the RKHS, 2) the inducing functions are not expressive enough to explain the prior. Both problems manifest themselves in an approximation that is overly smooth and over-estimates the predictive variance.

The Mercer representation of the kernel given in **??** implies that we have direct access to the Karhunen–Loève representation of the GP:

$$f(\boldsymbol{x}) = \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} \xi_{n,j} \sqrt{\lambda_n} ||\boldsymbol{x}|| \phi_{n,j}\left(\frac{\boldsymbol{x}}{||\boldsymbol{x}||}\right), \quad \text{where the } \xi_{n,j} \text{ are i.i.d. } \mathcal{N}(0,1). \tag{4.6}$$

Using this expression to compute the RKHS norm of a GP sample $f(\cdot)$ yields $||f||^2 = \sum_{n,j} \xi_{n,j}^2$, which is a diverging series. This is a clear indication that the GP samples do not belong to the RKHS [Kanagawa et al., 2018], and that expressions such as $\langle f(\cdot), g(\cdot)\rangle_{\mathcal{H}}$ should be manipulated with care. According to the definition given in **??**, the RKHS inner product is a series that converges if computed for any two elements $g(\cdot), h(\cdot) \in \mathcal{H}$. The inner product operator can however be extended to the case where one input lives in a space larger than $\mathcal{H}$, provided that restrictions are introduced on the second input to guarantee the convergence of the series. In

other words, even if the decay of the Fourier coefficients of $f(\cdot)$ is too slow to make it an element of $\mathcal{H}$, if the Fourier coefficients of $g(\cdot)$ decay quickly enough for the series $\sum_{n,j} \xi_{n,j} g_{n,j}/\sqrt{\lambda_n}$ to converge then $\langle f(\cdot), g(\cdot)\rangle_{\mathcal{H}}$ is well defined.

The above reasoning indicates that, for a given kernel $k(\cdot,\cdot)$, some activation functions $g_m(\cdot)$ will result in inducing variables $u_m = \langle f(\cdot), g_m(\cdot)\rangle_{\mathcal{H}}$ that are well defined whereas other activation functions do not. For example, if we consider the case of the Arc Cosine kernel and the ReLU inducing function, the decay rate of σ_n is proportional to the square root of λ_n [Bach, 2017; Bietti and Bach, 2020]. This implies that the inner product series diverges and that this kernel and inducing variable cannot be used together. Alternatively, using smoother activation functions for $\sigma(t)$, such as the Softplus, results in a faster decay of the coefficients σ_n and can guarantee that inducing variables are well defined.

An alternative to ensure the series convergence for any combination of kernel and activation function is to use a truncated approximation of the activation function $\tilde{g}_m(\cdot)$ where all the Fourier coefficients above a given level $\tilde{N}$ are set to zero, which basically turns the inner product series into a finite sum. **??** shows how the true and truncated activation functions for the ReLU and Softplus compare. These correspond to the orange functions in **??**, but are now plotted on a line. In the low to medium dimensional regime, we see that even for small truncation levels we approximate the ReLU and Softplus well. In higher dimensions this becomes more challenging for the ReLU.

Unexpressive inducing variables through truncation (??) The main concern with this truncation approach, however, comes from elsewhere: the larger $\tilde{N}$ is, the closer $\tilde{g}_m(\cdot)$ is to $g_m(\cdot)$, but the larger $||\tilde{g}_m(\cdot)||_{\mathcal{H}}$ becomes (to the point where it may be arbitrarily large). Similarly to ridge regression where the norm acts as a regulariser, using inducing functions with a large norm in SVGP models comes with a penalty which enforces more smoothness in the approximate posterior and limits its expressiveness. **??** shows how the norm of our ReLU inducing functions grow in the RKHS. So by making $\tilde{N}$ larger such that we approximate the ReLU better, we incur a greater penalty in the ELBO for using them. This leads to unexpressive inducing variables, which can be seen by the growing predictive uncertainty. The Softplus, which is part of the RKHS, does not suffer from this.

Unexpressive inducing variables through spectra mismatch (??) Any stationary kernel whose inputs $\boldsymbol{x}, \boldsymbol{x}' \in \mathbb{S}^{d-1}$ are restricted to the unit hypersphere is a zonal kernel (i.e., the kernel only depends on the dot-product). This means that we are not limited to only the Arc Cosine because we can replace the shape function in **??** by any stationary kernel, and our approach would still hold. For example, we could use the Matérn-5/2 shape function with $s_{\text{mat-5/2}}(t) = \left(1 + \sqrt{5}t + 5t^2/3\right)\exp\left(-\sqrt{5}t\right)$. However, in **??** we compare the fit of an SVGP model using a Matérn-5/2 kernel (left) to a model using an Arc Cosine kernel (right). While both models use our Softplus inducing variables, we clearly observe that the Matérn kernel

gives rise to a worse posterior model (lower ELBO and an over-estimation of the variance). In what follows we explain why this is the case.

4.2 Experiments

We have shown that that we can build SVGP models with basis functions that behave like neural net activations. This equivalence has the practical advantage that we can train the means of the GP layers in our DGP as if they are a neural network model — making use of the great advances made by the deep learning community. Once the mean of the DGP is trained, we can further optimise the remaining model hyper- and variational parameters w.r.t. the ELBO, which is a more principled objective [Fong and Holmes, 2019]. This approach allows us to exploit the benefit of both, the efficient training of the DNN in combination with the principled uncertainty estimate of the DGP. For all SVGP models we use the Arc Cosine kernel and inducing variables obtained with the Softplus activation (section 4.1.1) with $\tilde{N} = 20$, for the reasons explained in section 4.1.3.

The aim of the experiments is to highlight that (i) our method leads to valid inducing variables, (ii) our initialisation improves DGPs in terms of accuracy, and (iii) we are able to improve on Dropout-based [Gal and Ghahramani, 2016] neural networks in terms of calibrated uncertainty. We acknowledge that the NNs we benchmark against are the models for which we can build an equivalent DGP. While this leads to a fair comparison, it excludes recent improvements such as Transformer and Batch Norm layers.

From Neural Network to Activated DGP **??** shows the difference in predictive probability $p(y \,|\, \boldsymbol{x})$ for a DNN and our activated DGP, in the single-layer and three-layer case. We configure the models with a Softplus activation function and set the number of both inducing variables of the GP and hidden units of the DNN to 100. In this experiment, the first step is to optimise the DNN w.r.t. the binary cross-entropy objective, upon convergence we initialise the DGP with this solution and resume training of the DGP w.r.t. the ELBO. Especially in the single-layer case, we notice how the sharp edges from the NN are relaxed by the GP fit, and we see how the GP expresses uncertainty away from the data by letting $p(y \,|\, \boldsymbol{x}) \approx 0.5$. This is thanks to the ELBO, which balances both data fit and model complexity and simultaneously trains the uncertainty.

Regression on UCI benchmarks We compare a suit of models on a range of regression problems. The important aspect is that we keep the model configuration and training procedure fixed across datasets. We use three-layered models with 128 inducing variables or hidden units. In each layer, the number of output heads is equal to the input dimensionality of the data. The Activated DGP (ADGP) and neural network approaches (Vanilla and Dropout) use Softplus activation functions. The Dropout model [Gal and Ghahramani, 2016] uses a rate of 0.1 during

train and test, and the Vanilla model is a deterministic neural network that uses the training MSE as the empirical variance during prediction.

The DGP and ADGP both use the Arc Cosine kernel. The main difference is that the DGP has standard inducing points $u_m = f(\boldsymbol{z}_m)$, whereas ADGP makes use of our activated inducing variables $u_m = \langle f, g_m \rangle_{\mathcal{H}}$. The ADGP is trained in two steps: we first train the mean of the approximate posterior w.r.t. the MSE, and then optimise all parameters w.r.t. the ELBO.

In **??** we see that in general our ADGP model is more accurate than its neural network initialisation (Vanilla model) in terms of RMSE. This is a result of the second stage of training in which we use the ELBO rather than the MSE, which is especially beneficial to prevent overfitting on the smaller datasets. When it comes to NLPD, ADGP shows improvements over its NN initialisation for 5 datasets out of 7 and consistently outperforms classic DGPs.

Large scale image classification In this experiment we measure the performance of our models under dataset shifts [Ovadia et al., 2019]. For MNIST and Fashion-MNIST the out-of-distribution (OOD) test sets consist of rotated digits — from 0°(i.e. the original test set) to 180°. For CIFAR-10 we apply four different types of corruption to the test images with increasing intensity levels from 0 to 5. For MNIST and FASHION-MNIST the models consist of two convolutional and max-pooling layers, followed by two dense layers with 128 units and 10 output heads. The dense layers are either fully-connected neural network layers using a Softplus activation function (Vanilla and Dropout), or our Activated GP layers using the Arc Cosine kernel and Softplus inducing variables (ADGP). For the CIFAR-10 models, we use the residual convolutional layers from a ResNet [He et al., 2016] to extract useful features before passing them to our dense GP or NN layers. Details of the model architectures are given in **??**. As previously, the ADGP model is initialised to the solution of the Vanilla model, and training is then continued using the ELBO. In **??** we observe that the models perform very similar in terms of prediction accuracy, but that ADGP better account for uncertainty as evidenced by the Test Log Likelihood metric.

Chapter 5

Future Research

5.1 Non-Parametric Sequential Decision Making in Non-Euclidian Spaces

To date, my research has focussed on pushing Gaussian processes (GPs) into the realm of deep learning. I have developed deep convolutional Gaussian processes that mimic the convolutional and layered architectures encountered in many deep learning models [Dutordoir et al., 2020b], Conditional density estimation models which are similar to Variational Auto Encoders (VAEs) [Dutordoir et al., 2018], and more recently, deep Gaussian processes that have basis functions that are similar to neural network activation functions [Dutordoir et al., 2021a]. Through this line of work, I believe that we have showed that (deep) Gaussian processes are an interesting non-parametric alternative to Bayesian neural networks. TODO: Curve fitting

In what comes next, I want to separate the different regimes in which neural networks and Gaussian processes operate and thrive. For example, NNs can handle—and in practice require—very large datasets, whereas Gaussian process models are more comfortable in the small data regime. Neural networks, in the presence of large datasets, are extremely good at learning complex latent representations, as evidenced by the latest models in natural language processing and computer vision. Non-parametric Gaussian processes, on the contrary, work best on limited and noisy datasets. Datasets where each datapoint can be very expensive in terms of cost or time to obtain. I believe that this is a setting—in the era of deep learning—that has been under studied and valued, yet of high importance for many scientific or commercial applications.

Many real-world problems can be described as inferring properties of an expensive black-box function $f : \mathcal{X} \to \mathcal{Y}$, subject to a computational budget of T function evaluations. Typical examples are (global) optimisation, finding a level set (i.e. the set of points in $X \subset \mathcal{X}$ for which $f(x) > C, \forall x \in X$), or finding the shortest path between two nodes in a graph. In the graph example, the black-box function f would return the cost of traversing an edge and query the

cost of an edge would be very expensive. Naively applying Dijkstra would require the evaluation of f at every edge and thus potentially grossly exceeding the given budget of T evaluations.

Problem definition - Actions - Function

- Contextual multi-arm - Bayesian optimisation can be cast - multi arm setting is bayesian optimisation with discrete inputs

Gaussian processes as surrogate model in non-Euclidian spaces We want to place a *surrogate model* between the expensive function f and the algorithm. We model $\tilde{f}$ by a Gaussian process

$$\tilde{f} \sim \mathcal{GP} \tag{5.1}$$

1. Low dimensions
2. Prior knowledge
3. Limited, noisy and very expensive data
4. Non-Euclidian spaces: graphs, meshes, and closed manifolds (e.g. circle).

Basically settings where DNNs are never going to be competitive with GPs - low-dim, very data-efficient, high-cost - not even if someone figures out how to do DNN uncertainty right, due to GP regret guarantees (under reasonable assumptions) matching the best possible regret achievable by any model/decision system.

Related areas

1. Probabilistic numerics [Tubingen Manifesto, Osborne and Henning]. They are less interested in closing the loop and making decisions atop of the models. They usually plot the error bars on the estimator as their final result.
2. Bayesian optimisation methods. Special case.

Real-world problems

Graphs Social networks. Search for cliques or shortest paths.

Meshes Aerospace and civil engineering problems. “General” sensor placement.

Manifold Sphere. Interesting problem in astrophysics: when a gravitational wave detection is made there’s usually a very large uncertainty of its origin so optical telescopes have to sweep the sky looking for the source.

Objectives

1. Theory and analysis
2. Show the excellence of Gaussian processes in this regime
3. Benchmarks for future methods

Practical

1. Collaborators
   (a) Alex Terenin (Imperial College London), currently with Marc Deisenroth but starting post-doc at Cambridge.
   (b) Willie Neiswanger (Stanford University), with prof. Stefano Ermon.
2. We need to find the right problems to solve, which can take time.

5.2 On-going Projects

1. "Pay Attention to Deep Gaussian Processes"
   Transformer Layer Gaussian Processes using an explicit feature representation of the attention operation. The inducing points would play the role of keys in the attention layer.
   $$\exp(\boldsymbol{x}^\top \boldsymbol{y}) = \Phi^\top(\boldsymbol{x})\Phi(\boldsymbol{y})$$
2. Green's Inducing Functions for GPs in Balls
   (a) Indcing points: adaptivity
   (b) Variational Fourier Features: boundary conditions, doesn't scale with dimensionality
   (c) Variational Orthogonal Features:
   (d) Spherical Harmonics: restricted in kernel, diagonal Kuu
3. VISH-PI: Probabilistic Integration with Variational Inducing Spherical Harmonics, with Michael Osborne and Saad Hamid (student)

References

Nachman Aronszajn (1950). "Theory of reproducing kernels". In: *Transactions of the American mathematical society*.

Francis Bach (2017). "Breaking the Curse of Dimensionality with Convex Neural Networks". In: *Journal of Machine Learning Research*.

Matthias Bauer, Mark van der Wilk, and Carl Edward Rasmussen (2016). "Understanding probabilistic sparse Gaussian process approximations". In: *Advances in Neural Information Processing Systems 29 (NIPS 2016)*.

Alberto Bietti and Francis Bach (2020). "Deep Equals Shallow for ReLU Networks in Kernel Regimes". In: *arXiv preprint arXiv:2009.14397*.

Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra (2015). "Weight Uncertainty in Neural Network". In: *Proceedings of The 32nd International Conference on Machine Learning (ICML)*.

Thang D. Bui, Josiah Yan, and Richard E. Turner (2017). "A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation". In: *Journal of Machine Learning Research*.

David R Burt, Carl Edward Rasmussen, and Mark van der Wilk (2020). "Variational orthogonal features". In: *arXiv preprint arXiv:2006.13170*.

David R. Burt, Carl E. Rasmussen, and Mark van der Wilk (2019). "Rates of Convergence for Sparse Variational Gaussian Process Regression". In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Youngmin Cho and Lawrence K. Saul (2009). "Kernel Methods for Deep Learning". In: *Advances in Neural Information Processing Systems 22 (NIPS)*.

Kurt Cutajar, Edwin V. Bonilla, Pietro Michiardi, and Maurizio Filippone (2017). "Random feature expansions for deep Gaussian processes". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*.

Feng Dai and Yuan Xu (2013). *Approximation Theory and Harmonic Analysis on Spheres and Balls*. Springer.

Andreas Damianou and Neil D. Lawrence (2013). "Deep Gaussian Processes". In: *Proceedings of the 16th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Justin Domke and Daniel R Sheldon (2018). "Importance weighting and variational inference". In: *Advances in Neural Information Processing Systems 31*.

Vincent Dutordoir, Nicolas Durrande, and James Hensman (2020a). "Sparse Gaussian Processes with Spherical Harmonic Features". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Vincent Dutordoir, James Hensman, Mark van der Wilk, Carl Henrik Ek, Zoubin Ghahramani, and Nicolas Durrande (2021a). "Deep Neural Networks as Point Estimate for Deep Gaussian Processes". In: *submission to NeurIPS*.

Vincent Dutordoir, Hugh Salimbeni, Eric Hambro, John McLeod, Felix Leibfried, Artem Artemev, Mark van der Wilk, James Hensman, Marc P Deisenroth, and ST John (2021b). "GPflux: A Library for Deep Gaussian Processes". In: *Proceedings of the 3th International Conference on Probabilistic Programming (2021)*.

Vincent Dutordoir, Hugh Salimbeni, James Hensman, and Marc Deisenroth (2018). "Gaussian process conditional density estimation". In: *Advances in Neural Information Processing Systems*.

Vincent Dutordoir, Mark van der Wilk, Artem Artemev, and James Hensman (2020b). "Bayesian Image Classification with Deep Convolutional Gaussian Processes". In: *Proceedings of the 23th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

David Duvenaud, Oren Rippel, Ryan Adams, and Zoubin Ghahramani (2014). "Avoiding pathologies in very deep networks". In: *Proceedings of the 17th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Costas Efthimiou and Christopher Frye (2014). *Spherical Harmonics in p Dimensions*. World Scientific Publishing.

Edwin Fong and Chris Holmes (2019). "On the marginal likelihood and cross-validation". In: *arXiv preprint arXiv:1905.08737*.

Andrew Y. K. Foong, David R. Burt, Yingzhen Li, and Richard E. Turner (2020). "On the Expressiveness of Approximate Inference in Bayesian Neural Networks". In: *Advances in Neural Information Processing Systems 33 (NeurIPS)*.

Yarin Gal and Zoubin Ghahramani (2016). "Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning Zoubin Ghahramani". In: *Proceedings of The 33rd International Conference on Machine Learning (ICML)*.

Adrià Garriga-Alonso, Carl E. Rasmussen, and Laurence Aitchison (2018). "Deep Convolutional Networks as shallow Gaussian Processes". In: *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*.

James Hensman, Nicolas Durrande, and Arno Solin (2018). "Variational Fourier Features for Gaussian Processes". In: *Journal of Machine Learning Research*.

James Hensman, Nicolo Fusi, and Neil D. Lawrence (2013). "Gaussian Processes for Big Data". In: *Proceedings of the 29th Conference on Uncertainty in Artificial Intelligence (UAI)*.

James Hensman and Neil D. Lawrence (2014). "Nested Variational Compression in Deep Gaussian Processes". In: *arXiv preprint arXiv:1412.1370*.

James Hensman, Alexander G. de G. Matthews, and Zoubin Ghahramani (2015). "Scalable Variational Gaussian Process Classification". In: *Proceedings of the 18th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Jiri Hron, Alexander G. de G. Matthews, and Zoubin Ghahramani (2018). "Variational Bayesian dropout: pitfalls and fixes". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*.

Arthur Jacot, Franck Gabriel, and Clément Hongler (2018). "Neural Tangent Kernel: Convergence and Generalization in Neural Networks". In: *Advances in Neural Information Processing Systems 31 (NeurIPS)*.

Motonobu Kanagawa, Philipp Hennig, Dino Sejdinovic, and Bharath K Sriperumbudur (2018). "Gaussian Processes and Kernel Methods: A Review on Connections and Equivalences". In: *arXiv preprint arXiv:1807.02582*.

Durk Kingma, Tim Salimans, and Max Welling (2015). "Variational Dropout and the Local Reparameterization Trick". In: *Advances in Neural Information Processing Systems 28 (NIPS)*.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton (2012). "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25 (NIPS)*.

Malte Kuss and Carl E. Rasmussen (2005). "Assessing Approximate Inference for Binary Gaussian Process Classification". In: *Journal of Machine Learning Research*.

Serge Lang (1993). *Real and Functional Analysis*. Springer.

Miguel Lázaro-Gredilla and Aníbal Figueiras-Vidal (2009). "Inter-domain Gaussian Processes for Sparse Inference using Inducing Features". In: *Advances in Neural Information Processing Systems 22 (NIPS)*.

Felix Leibfried, Vincent Dutordoir, S. T. John, and Nicolas Durrande (2020). "A Tutorial on Sparse Gaussian Processes and Variational Inference". In: *arXiv preprint arXiv:2012.13962*.

David J. C. MacKay (1992a). "A Practical Bayesian Framework for Backpropagation Network". In: *Neural Computation*.

David J. C. MacKay (1992b). "Bayesian Model Comparison and Backprop Nets". In: *Advances in Neural Information Processing Systems 4 (NIPS 1991)*.

David J. C. MacKay (1998). "Choice of Basis for Laplace Approximation". In: *Machine Learning*.

Alexander G. de G. Matthews, James Hensman, Turner E. Richard, and Zoubin Ghahramani (2016). "On Sparse Variational Methods and the Kullback-Leibler Divergence between Stochastic Processes". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Alexander G. de G. Matthews, Mark Rowland, Jiri Hron, Richard E. Turner, and Zoubin Ghahramani (2018). "Gaussian process behaviour in wide deep neural networks". In: *Proceedings of the 6th International Conference on Learning Representations (ICLR)*.

Thomas P. Minka (2001). "Expectation propagation for approximate Bayesian inference". In: *Proceedings of the Seventeenth conference on Uncertainty in Artificial Intelligence (UAI 2001)*. Morgan Kaufmann Publishers Inc.

Radford M. Neal (1992). "Bayesian Mixture Modeling". In: *Maximum Entropy and Bayesian Methods*.

Radford M. Neal (1995). *Bayesian Learning for Neural Networks*. Springer.

Roman Novak, Lechao Xiao, Jaehoon Lee, Yasaman Bahri, Greg Yang, Jiri Hron, Daniel A. Abolafia, Jeffrey Pennington, and Jascha Sohl-Dickstein (2019). "Bayesian deep convolutional networks with many channels are Gaussian processes". In: *Proceedings of the 7th International Conference on Learning Representations (ICLR)*.

Yaniv Ovadia, Emily Fertig, Jie Ren, Zachary Nado, D. Sculley, Sebastian Nowozin, Joshua Dillon, Balaji Lakshminarayanan, and Jasper Snoek (2019). "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift". In: *Advances in Neural Information Processing Systems 32 (NeurIPS)*.

John A. Peacock (1999). *Cosmological Physics*. Cambridge University Press.

Florent Perronnin, Yan Liu, Jorge Sánchez, and Hervé Poirier (2010). "Large-scale image retrieval with compressed fisher vectors". In: *2010 IEEE computer society conference on computer vision and pattern recognition*. IEEE.

Joaquin Quiñonero-Candela and Carl E. Rasmussen (2005). "A Unifying View of Sparse Approximate Gaussian Process Regression". In: *Journal of Machine Learning Research*.

Ali Rahimi and Benjamin Recht (2008). "Random Features for Large-Scale Kernel Machines". In: *Advances in Neural Information Processing Systems 20 (NIPS 2007)*.

Carl E. Rasmussen and Christopher K. I. Williams (2006). *Gaussian Processes for Machine Learning*. MIT Press.

Tim G. J. Rudner, Dino Sejdinovic, and Yarin Gal (2020). "Inter-domain Deep Gaussian Processes". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*.

Hugh Salimbeni and Marc P. Deisenroth (2017). "Doubly Stochastic Variational Inference for Deep Gaussian Processes". In: *Advances in Neural Information Processing Systems 30 (NIPS)*.

Edward Snelson and Zoubin Ghahramani (2005). "Sparse Gaussian Processes using Pseudo-inputs". In: *Advances in Neural Information Processing Systems 4 (NIPS 2005)*.

Shengyang Sun, Jiaxin Shi, and Roger B. Grosse (2021). "Neural Networks as Inter-Domain Inducing Points". In: *3rd Symposium on Advances in Approximate Bayesian Inference*.

Michalis K. Titsias (2009). "Variational Learning of Inducing Variables in Sparse Gaussian Processes". In: *Proceedings of the 12th International Conference on Artificial Intelligence and Statistics (AISTATS)*.

Mark van der Wilk, Matthias Bauer, S. T. John, and James Hensman (2018). "Learning Invariances using the Marginal Likelihood". In: *Advances in Neural Information Processing Systems 31 (NeurIPS)*.

Mark van der Wilk, Vincent Dutordoir, S. T. John, Artem Artemev, Vincent Adam, and James Hensman (2020). "A Framework for Interdomain and Multioutput Gaussian Processes". In: *arXiv preprint arXiv:2003.01115*.

Mark van der Wilk, Carl E. Rasmussen, and James Hensman (2017). "Convolutional Gaussian Processes". In: *Advances in Neural Information Processing Systems 30 (NIPS)*.

Ke Wang, Geoff Pleiss, Jacob Gardner, Stephen Tyree, Kilian Q. Weinberger, and Andrew Gordon Wilson (2019). "Exact Gaussian Processes on a Million Data Points". In: *Advances in Neural Information Processing Systems 32 (NeurIPS)*.

Holger Wendland (2005). *Scattered Data Approximation*. Cambridge University Press.

Christopher K. I. Williams (1998). "Computing with Infinite Networks". In: *Advances in Neural Information Processing Systems 11 (NIPS 1997)*.