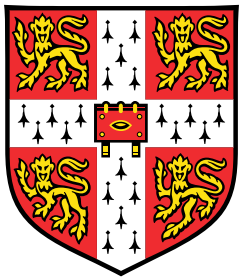
Non-Parameteric Decision Making in Non-Euclidean Spaces



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

1 Introduction **1**

2 Theoretical Framework **3**

2.1 Gaussian Processes . . . 3

2.1.1 The Beauty of Gaussian Process Regression: Exact Bayesian Inference . 4

2.2 Approximate Inference with Sparse Gaussian Processes . . . 6

2.2.1 Interdomain Inducing Variables . . . 8

2.3 A Kernel Perspective on Gaussian Processes . . . 10

2.3.1 Spectral Formulation . . . 12

3 Linear Time Gaussian Processes on Hyperspheres **15**

3.1 Mercer Representation of Dot-Product Kernels . . . 15

3.2 Variational Spherical Harmonic Gaussian Processes . . . 18

3.3 Homogeneous Extension to $\mathbb{R}^d$ . . . 19

3.3.1 First-order Arc Cosine kernel . . . 20

3.4 Relation to Variational Fourier Features . . . 21

3.5 Experiments . . . 22

3.5.1 Toy Experiment: Banana Classification . . . 22

3.5.2 Large-Scale Regression on Airline Delay . . . 23

3.6 Concluding Remarks . . . 24

4 Deep Neural Networks as Point Estimates for Deep Gaussian Processes **25**

4.1 Related Work . . . 25

4.2 Deep Gaussian Processes . . . 27

4.2.1 Connection to Deep Neural Networks . . . 27

4.3 Gaussian Process Layers with Activated Inducing Variables . . . 28

4.3.1 Activated Inducing Functions . . . 29

4.3.2 Activated Interdomain Inducing Variables . . . 30

4.3.3 Analysis of the interplay between kernels and inducing functions . . . 31

4.4 Experiments . . . 34

4.4.1 Regression on UCI benchmarks . . . 34
4.4.2 Large scale image classification . . . 35
4.5 Conclusion . . . 35

5 Future Research Agenda **37**
5.1 Introduction . . . 37
5.2 Markov Decision Process . . . 38
5.2.1 A Model-based Approach . . . 39
5.3 Gaussian Processes as Surrogate Models in non-Euclidian manifolds . . . 40
5.4 Research Agenda . . . 41
5.4.1 Adopotion: Geometric Kernel Package . . . 42
5.4.2 Efficiency: Sparse Manifold Gaussian processes . . . 42
5.4.3 Application: Sequential decision-making on non-Euclidian spaces . . . 43

Appendix A Spherical Harmonics on Hyperspheres **51**
A.1 Algorithm: Zonal Spherical Harmonics . . . 53

Appendix B Analytic computation of eigenvalues for zonal functions **57**
B.1 Arc Cosine kernel . . . 57
B.2 ReLU activation function . . . 60

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 **??**, 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*

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, 2006b] 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, 2006b].

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; T. D. 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:

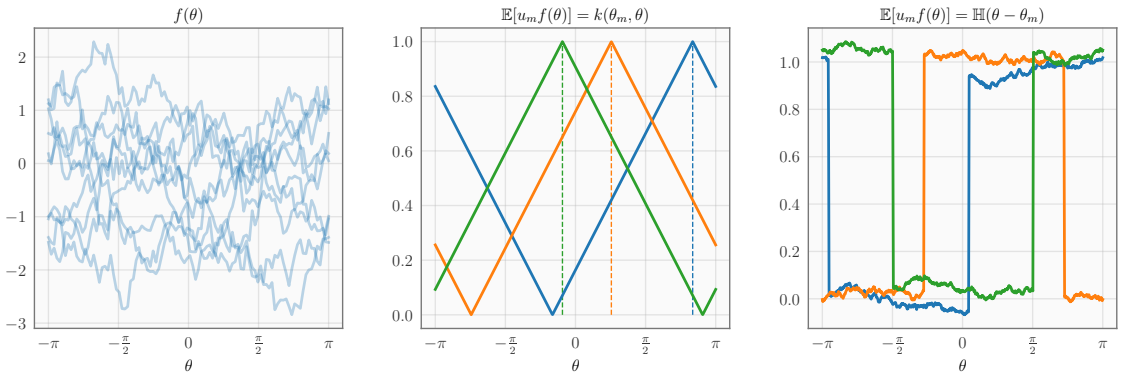
$$u_m = \langle f, \psi_m \rangle_{\mathcal{H}},$$

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 A Kernel Perspective on Gaussian Processes

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

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

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

$$\begin{aligned}
\mathrm{Cov}(f(x), f(x')) &= \sum_{n,m} \sqrt{\lambda_m} \sqrt{\lambda_n} \mathbb{E}[\xi_n \xi_m] \phi_n(x) \phi_m(x') && \textit{eq. (2.34)} && (2.35)\\
&= \sum_n \lambda_n \phi_n(x) \phi_n(x') && \mathbb{E}[\xi_n \xi_m] = \delta_{n\,m} && (2.36)\\
&= k(x, x') && \textit{eq. (2.30)}. && (2.37)
\end{aligned}$$

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 Hyperspheres

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 $\mathcal{K}$ commutes with the Laplace-Beltrami operator $\Delta^{\mathbb{S}^{d-1}}$, henceforth referred to as simply the Laplacian or the 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, the diagonalisation of the Laplace operator on $\mathbb{S}^{d-1}$ is a well-studied problem. In the following paragraph we first proof the commutativity of the operators before covering the definition of the RKHS.

Commutativity of $\mathcal{K}$ and $\Delta^{\mathbb{S}^{d-1}}$. We now proof that the Laplacian and the kernel operator of zonal kernels commute. For this, we first show that for zonal kernels and $x, s \in \mathbb{S}^{d-1}$ the following property holds

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

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

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

Relying on integration by parts and the previous result, we obtain

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

$$= \int_{\mathbb{S}^{d-1}} f(x)\Delta_x^{\mathbb{S}^{d-1}} k(x,x')\mathrm{d}\nu(x) \qquad 2 \times \text{Integration by parts} \tag{3.12}$$

$$= \int_{\mathbb{S}^{d-1}} f(x)\Delta_{x'}^{\mathbb{S}^{d-1}} k(x,x')\mathrm{d}\nu(x) \qquad \text{eq. (3.10)} \tag{3.13}$$

$$= \Delta^{\mathbb{S}^{d-1}}[\mathcal{K}f] \tag{3.14}$$

which shows that the two operators commute and in turn implies that they 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. The spherical harmonics $\phi_{n,j}$ form an orthonormal basis for the square-integrable functions on the hypersphere. They are indexed with a level n and an index within each level $j \in \{1, \ldots, N_n^d\}$. See chapter A for a comprehensive introduction to spherical harmonics, as well as a practical algorithm to compute them for relatively large d. Figure 3.1 shows the first 4 levels of spherical harmonics in $\mathbb{R}^3$.

The above reasoning about the commutativity of the Laplace-Beltrami and kernel operator and the diagonalisation of the Laplace-Beltrami operator by the spherical harmonics, can be summarised by the following theorem:

Theorem 4 (Mercer decomposition)**.** *Any zonal kernel $k(x,x') = \kappa(x^\top x')$ on the hypersphere can be decomposed as*

$$k(x,x') = \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} \lambda_n \phi_{n,j}(x)\phi_{n,j}(x'), \tag{3.15}$$

where $x, x' \in \mathbb{S}^{d-1}$ and λ_n are positive coefficients, $\phi_{n,j}$ denote the elements of the spherical harmonic basis in $\mathbb{S}^{d-1}$, and N_n^d corresponds to the number of spherical harmonics for a given level n. As a result of the Funk-Hecke theorem, the associated eigenvalues only depending on the level n

$$\lambda_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{-1}^{1} \kappa(t)\, C_n^{(\alpha)}(t)\, (1-t^2)^{\frac{d-3}{2}} \mathrm{d}t, \tag{3.16}$$

where $C_n^{(\alpha)}$ is the Gegenbauer polynomial of degree n, $\alpha = \frac{d-2}{2}$, ω_d is a constant that depends on the surface area of the hypersphere. Analytic expressions are given in chapter A.

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). Given the Mercer decomposition, we can equivalently, define the associated

RKHS as

$$\mathcal{H} = \left\{ f = \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} f_{n,j}\phi_{n,j} : ||f||_{\mathcal{H}} < \infty \right\}, \quad \text{where} \quad \langle g, h \rangle_{\mathcal{H}} = \sum_{n,j} \frac{g_{n,j} h_{n,j}}{\lambda_n} \tag{3.17}$$

is the *reproducing* inner product between two functions $g, h \in \mathcal{H}$. The reproducing property of the RKHS implies that for $f \in \mathcal{H}$: $f(x) = \langle f, k(x, \cdot) \rangle_{\mathcal{H}}$, which will enable the constuction of spherical harmonic inducing features of discussing in the following section.

3.2 Variational Spherical Harmonic Gaussian Processes

We can now build on the results from the previous section, combined with interdomain inducing variables (see section 2.2.1), to propose a novel sparse variational GP model. Specifically, we are using interdomain inducing variables that are defined through the zonal kernel's RKHS inner product. As a result we will obtain expressive features $\mathbf{k}_{\boldsymbol{u}\cdot}$ that exhibit non-local influence, and efficient inducing variables that induce diagonal structure in $\mathbf{K}_{\boldsymbol{uu}}$. We achieve this by defining the inducing variables u_m to be the inner product between the GP and spherical harmonics:[1]

$$u_m = \langle f, \phi_m \rangle_{\mathcal{H}} \quad \text{for} \quad m \in \{1, \ldots, M\}. \tag{3.18}$$

To leverage these new inducing variables we need to compute two quantities: 1) the covariance between u_m and f for $\mathbf{k}_{\boldsymbol{u}\cdot}$, and 2) the covariance between the inducing variables themselves for the $\mathbf{K}_{\boldsymbol{uu}}$ matrix. Firstly, we compute the covariance of the inducing variables and the GP

$$\begin{aligned}
[\mathbf{k}_{\boldsymbol{u}\cdot}]_m &= \mathbb{E}[f(\cdot)\, u_m] && \text{Definition covariance} && (3.19)\\
&= \mathbb{E}[f(\cdot)\, \langle f, \phi_m \rangle_{\mathcal{H}}] && \text{Definition } u_m && (3.20)\\
&= \langle k(\cdot, \cdot), \phi_m \rangle_{\mathcal{H}} && k(\cdot, \cdot) = \mathbb{E}[f(\cdot) f] && (3.21)\\
&= \phi_m(\cdot) && \text{Reproducing property.} && (3.22)
\end{aligned}$$

where we relied on the linearity of the expectation and the inner product and the reproducing property of $\mathcal{H}$. Interestingly, using this construction we obtain the RKHS basis functions $\{\phi_m\}$ as the features for the sparse approximation. Secondly, the covariance between the inducing variables is given by

$$[\mathbf{K}_{\boldsymbol{uu}}]_{m,m'} = \mathbb{E}[u_m\, u_{m'}] = \langle \phi_m, \phi_{m'} \rangle_{\mathcal{H}} = \frac{\delta_{mm'}}{\lambda_m}, \tag{3.23}$$

where $\delta_{mm'}$ is the Kronecker delta. Crucially, this means that $\mathbf{K}_{\boldsymbol{uu}}$ is a diagonal matrix $\text{diag}(\lambda_1, \lambda_2, \ldots, \lambda_M)^{-1}$ containing the inverse eigenvalues on its diagonal. Finally, substituting

[1] Note that in the context of inducing variables, we switch to a single integer m to index the spherical harmonics and order them first by increasing level n, and then by increasing j within a level.

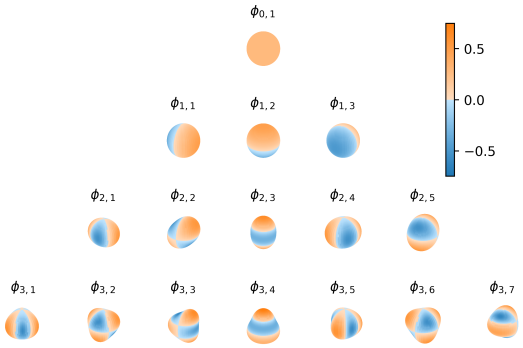


Fig. 3.1 Spherical Harmonics on $\mathbb{S}^2$

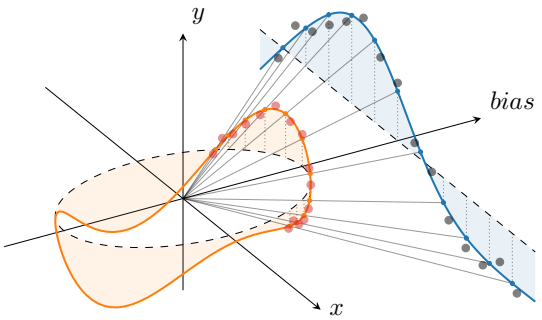


Fig. 3.2 Example Homogeneous Extension

$\mathbf{K}_{\boldsymbol{uu}}$ and $\mathbf{k}_{\boldsymbol{u}\cdot}$ into the sparse variational approximation (eq. (2.13)) leads to

$$q(f) = \mathcal{GP}\Big(\tilde{\boldsymbol{\phi}}^\top(x)\,\boldsymbol{m};\ k(x,x') + \tilde{\boldsymbol{\phi}}^\top(x)(\mathbf{S} - \mathbf{K}_{\boldsymbol{uu}})\tilde{\boldsymbol{\phi}}(\boldsymbol{x}')\Big), \qquad \text{with} \quad \tilde{\boldsymbol{\phi}}(x) = [\lambda_m \phi_m(x)]_{m=1}^M \in \mathbb{R}^M. \tag{3.24}$$

Through this construction we created a SVGP model with 1) spherical harmonic basis functions, and 2) a diagonal $\mathbf{K}_{\boldsymbol{uu}}$ matrix, which means that we do not incur the cubic cost of matrix inversion in the evaluation of the model or the variational lower bound. The sperical harmonic basis functions have as advantage that they have global support, so that even a few of them are able to capture the rough shape of the function. This is similar to the Fourier series approximation, in which the first few terms are already able to roughly approximate the function. In the experiments we will discuss this phenomenon in more detail.

Note that while the inducing variables $\{u_m\}$ are independent under the prior (i.e. $p(\boldsymbol{u}) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\boldsymbol{uu}})$ with $\mathbf{K}_{\boldsymbol{uu}}$ a diagonal matrix), that does not imply that the posterior over the inducing variables is diagonal as well. As a result, we still parameterise $q(\boldsymbol{u}) = \mathcal{N}(\boldsymbol{m}, \mathbf{S})$, where $\mathbf{S}$ is a full-rank matrix which captures all the pairwise covariances.

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

So far, the domain of interest has been the hypersphere $\mathbb{S}^{d-1}$. However, most regression problems in machine learning have inputs defined on the complete Euclidean space. We can address this problem as follows. Consider $x \in \mathbb{R}^{d-1}$ to be the input of the dataset and $y \in \mathbb{R}$ the corresponding output.

1. We start by concatenating the datapoint $x \in \mathbb{R}^{d-1}$ with a bias term $b \in \mathbb{R}$ to obtain $x = [x, b] \in \mathbb{R}^d$. This operation corresponds to embedding the (d–1)-dimensional datapoint on a plane in $\mathbb{R}^d$ where $x_d = b$.

2. Subsequently, the data is linearly projected to the hypersphere through normalisation: $x = \frac{x}{||x||} \in \mathbb{S}^{d-1}$ and $y = \frac{y}{||x||}$.

3. Based on the projected data we learn $f \sim \mathcal{GP}$ on $\mathbb{S}^{d-1}$ using the approach outlined in the previous section.

4. Finally, the function on the sphere can be extended to the whole domain by a homogeneous extension thereof: $g : \mathbb{R}^d \to \mathbb{R}$ with $g(x) = ||x|| f(\frac{x}{||x||})$. In other words, by linearly extrapolating the values on the hypersphere we obtain the values of the function in the data plane.

This procedure is explained in fig. 3.2, which gives an illustration of the mapping between a 2D dataset (grey dots) embedded into a 3D space and its projection (orange dots) onto the unit half-circle using a linear mapping.

3.3.1 First-order Arc Cosine kernel

Although this setup may seem very arbitrary, it is inspired by the important works on the limits of neural networks as Gaussian processes. The first order Arc Cosine kernel [Cho and Saul, 2009] is one of the most well-known examples, and mimics the computation of infinitely wide fully connected layers with ReLU activations. Let $\sigma(t) = \max(0, t)$, then the covariance between function values of $f(x) = \sigma(w^\top x)$ for $w \sim \mathcal{N}\left(0, d^{-1/2}\mathbf{I}_d\right)$ and $w \in \mathbb{R}^d$ is given by

$$k(x, x') = \mathbb{E}_w\Big[\sigma(w^\top x)\,\sigma(w^\top x')\Big] = \underbrace{||x||||x'||}_{\text{radial}}\ \underbrace{\frac{1}{\pi}\left(\sqrt{1-t^2} + t\,(\pi - \arccos t)\right)}_{\text{angular (shape function) } \kappa(t)}, \tag{3.25}$$

where $t = \frac{\boldsymbol{x}^\top \boldsymbol{x}'}{||\boldsymbol{x}||||\boldsymbol{x}'||}$. Indeed, we observe how the first order Arc Cosine kernel factorises in a radial and an angular component. The radial component simply linearly extrapolates the values on the hypersphere, while the angular component encodes the covariance on the hypersphere and is only a function of the geodestic distance. This factorisation exactly matches our setup. We can generalise the zonal kernels of the previous section and their corresponding RKHS to the complete domain $\mathbb{R}^d$ using the following definition

$$k(x, x') = ||x||||x'||\kappa\left(\frac{x^\top x'}{||x||||x'||}\right), \tag{3.26}$$

which leads to an RKHS consisting of functions defined on $\mathbb{R}^d$ of the form $g(x) = ||x||\, f(\frac{x}{||x||})$, where $f \in \mathcal{H}$ (from eq. (3.17)) is defined on the unit hypersphere but fully determines the function on $\mathbb{R}^d$. Straightforwardly we can extend the Mercer decomposition of theorem 4 to

$$k(x, x') = \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} \lambda_n ||x|| \phi_{n,j}(x) ||x'|| \phi_{n,j}(x'), \tag{3.27}$$

where the eigenvalues λ_n and reproducing inner-product remain unchanged and are given by eq. (3.16) and eq. (3.17), respectively.

3.4 Relation to Variational Fourier Features

The most closely related method to the presented method, which going forward we are going to refer to as Variational Inducing Spherical Harmonics (VISH), is the Variational Fourier Feature (VFF) approach proposed by Hensman et al. [2018]. VFF 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:

$$u_m = \langle f, \psi_m \rangle_{\mathcal{H}}, \tag{3.28}$$

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

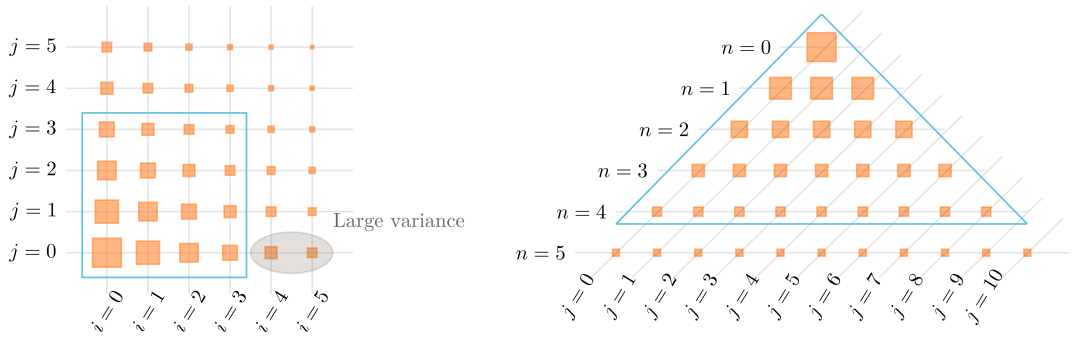
$$\mathbf{K}_{\boldsymbol{uu}} = [\langle \psi_i, \psi_j \rangle_{\mathcal{H}}]_{i,j=0}^{M-1} \qquad \text{and} \qquad \mathbf{k}_{\boldsymbol{u}\cdot} = [\psi_i(x)]_{i=0}^{M-1}. \tag{3.29}$$

This results in several advantages. First, the features $\mathbf{k}_{\boldsymbol{u}\cdot}$ 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. Finally, the variance of the inducing variables typically decays quickly with increasing frequencies, which means that by selecting the first M elements of the Fourier basis we pick the features that carry the most signal.

The main flaw of VFF comes from the way it generalises to multidimensional input spaces. The approach in Hensman et al. [2018] for getting a set of d-dimensional inducing functions consists of taking the outer product of d univariate basis and to consider separable kernels so that the elements of $\mathbf{K}_{\boldsymbol{uu}}$ are given by the product of the inner products in each dimension. For example, in dimension 2, a set of M^2 inducing functions is given by $\{(x_1, x_2) \mapsto \psi_i(x_1)\psi_j(x_2)\}_{0 \leq i,j \leq M-1}$, and entries on $\mathbf{K}_{\boldsymbol{uu}}$ are $\langle \psi_i\psi_j, \psi_k\psi_l \rangle_{\mathcal{H}_{k_1 k_2}} = \langle \psi_i, \psi_k \rangle_{\mathcal{H}_{k_1}} \langle \psi_j, \psi_l \rangle_{\mathcal{H}_{k_2}}$. This construction scales poorly with the dimension: for example choosing a univariate basis as simple as $\{1, \cos, \sin\}$ for an eight-dimensional problem already results in more that 6,500 inducing functions. Additionally, this construction is very inefficient in terms of captured variance, as we illustrate in Figure 3.3a for a 2D input space. The figure shows that the prior variance associated with the inducing function $\psi_i(x_1)\psi_j(x_2)$ vanishes quickly when both i and j increase. This means that most of the inducing functions on which the variational posterior is built are irrelevant, whereas some important ones such as $\psi_i(x_1)\psi_0(x_2)$ or $\psi_0(x_1)\psi_j(x_2)$ for $i, j \geq \sqrt{M}$ are important but ignored. Although we used a 2D example to illustrate this poor behaviour, it is important to bear in mind that the issue gets exacerbated for higher dimensional input spaces.

On the contrary for VISH, given that we ordered the spherical harmonic by increasing level n, choosing the first M elements means we will select first inducing functions with low angular frequency (see fig. 3.1). Provided that the kernel spectral density is a decreasing function (this

will be true for classic covariances, but not for quasi-periodic ones), this means that the selected inducing variables correspond to the ones carrying the most signal according to the prior. In other words, the decomposition of the kernel can be compared to an infinite dimensional principal component analysis, and our choice of the inducing function is optimal since we pick the ones with the largest variance. This is illustrated in fig. 3.3b, which shows the analogue of fig. 3.3a for spherical harmonic inducing functions.



(a) VFF (b) VISH

Fig. 3.3 Illustration of the variance of the prior and the inducing variables when using VFF or VISH with a two dimensional input space. Each square corresponds to an inducing function $\psi_{i,j}$ (VFF) or $\phi_{n,j}$ (VISH) and the area of each square is proportional to the variance of the associated inducing variable. For a given number of inducing variables as highlighted with the blue line, VFF does not select the inducing variables that contribute the most to the prior. The selection in VISH on the other hand is optimal.

3.5 Experiments

We evaluate our method Variational Inference with Spherical Harmonics (VISH) on a regression and a classification problem to demonstrate that VISH performs competitively in terms of accuracy and uncertainty quantification, while also being extremely fast (modelling a dataset of 6 million 8D datapoints in less than 2 minutes on a standard desktop).

3.5.1 Toy Experiment: Banana Classification

The banana dataset is a 2D binary classification problem [Hensman et al., 2015]. In fig. 3.4 we show three different fits of VISH with $M \in \{9,\ 225,\ 784\}$ spherical harmonics, which correspond respectively to maximum levels of 2, 14, and 27 for our inducing functions. Since the variational framework provides a guarantee that more inducing variables must be monotonically better [Titsias, 2009], we expect that increasing the number of inducing functions will provide improved approximations. This is indeed the case as we show in the rightmost panel: with increasing M the ELBO converges and the fit becomes tighter.

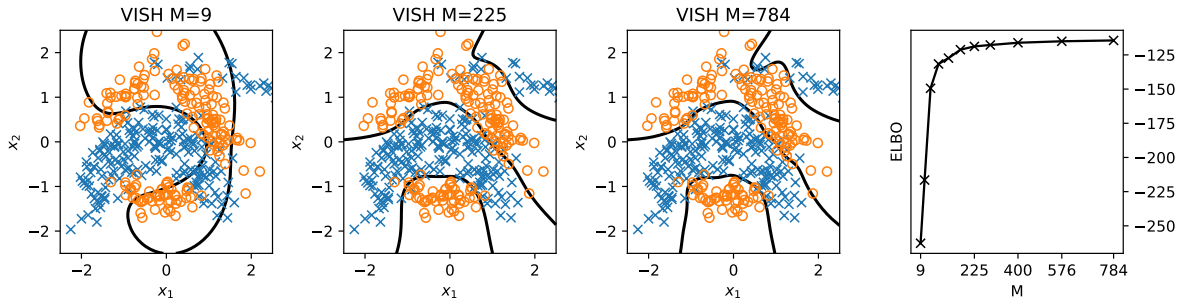


Fig. 3.4 Classification of the 2D banana dataset with growing number of spherical harmonic basis functions. The right plot shows the convergence of the ELBO with respect to increasing numbers of basis functions.

		$N = 10,000$			$N = 100,000$		$N = 1,000,000$		$N = 5,929,413$		
Method	M	MSE	NLPD	Time	MSE	NLPD	MSE	NLPD	MSE	NLPD	Time
VISH	210	0.91 ± 0.16	1.328 ± 0.09	1.86 ± 0.38	0.826 ± 0.052	1.28 ± 0.03	0.84 ± 0.01	1.29 ± 0.01	0.833 ± 0.004	1.29 ± 0.002	41.32 ± 0.81
VISH	660	0.90 ± 0.16	$\mathbf{1.326 \pm 0.09}$	4.76 ± 1.25	0.808 ± 0.052	$\mathbf{1.27 \pm 0.03}$	0.83 ± 0.03	$\mathbf{1.28 \pm 0.01}$	0.834 ± 0.055	$\mathbf{1.27 \pm 0.002}$	160.8 ± 3.80
A-VFF	30/dim.	$\mathbf{0.89 \pm 0.15}$	1.362 ± 0.09	6.78 ± 0.85	0.819 ± 0.05	1.32 ± 0.03	0.83 ± 0.01	1.33 ± 0.03	0.827 ± 0.004	1.32 ± 0.007	75.61 ± 0.75
SVGP	500	0.90 ± 0.16	1.358 ± 0.09	836.54 ± 0.78	$\mathbf{0.808 \pm 0.05}$	1.31 ± 0.03	$\mathbf{0.82 \pm 0.01}$	1.32 ± 0.002	$\mathbf{0.814 \pm 0.004}$	1.31 ± 0.002	918.77 ± 1.21

Table 3.1 Predictive mean squared errors (MSEs), negative log predictive densities (NLPDs) and wall-clock time in seconds with one standard deviation based on 10 random splits on the airline arrival delays experiment. Total dataset size is given by N and in each split we randomly select 2/3 and 1/3 for training and testing.

3.5.2 Large-Scale Regression on Airline Delay

This experiment illustrates three core capabilities of VISH: 1) it can deal with large datasets and 2) it is computationally and time efficient 3) the model improves performance in terms of NLPD.

We use the 2008 U.S. airline delay dataset to asses these capabilities. The goal of this problem is to predict the amount of delay y given eight characteristics x of a flight, such as the age of the aircraft (number of years since deployment), route distance, airtime, etc. We follow the exact same experiment setup as Hensman et al. [2018][2] and evaluate the performance on 4 datasets of size 10,000, 100,000, 1,000,000, and 5,929,413 (complete dataset), created by subsampling the original one. For each dataset we use two thirds of the data for training and one third for testing. Every split is repeated 10 times and we report the mean and one standard deviation of the MSE and NLPD. For every run the outputs are normalized to be a centered unit Gaussian and the inputs are scaled to be within $[-1, 1]$.

Table 3.1 shows the outcome of the experiment. The results for VFF and SVGP are from Hensman et al. [2018]. We observe that VISH improves on the other methods in terms of NLPD and is within error bars in terms of MSE. Given the variability in the data the GP models improve when more data is available during training.

[2] https://github.com/jameshensman/VFF

Given the dimensionality of the dataset, a full-VFF model is completely infeasible. As an example, using just four frequencies per dimension would already lead to $M = 4^8 = 65,536$ inducing variables. So VFF has to resort to an additive model with a prior covariance structure given as a sum of Matérn-3/2 kernels for each input dimension, as in **??**. Each of the functions f_d is approximated using 30 frequencies. We report two variants of VISH: one using all spherical harmonics up to degree 3 (M=210) and another up to degree 4 (M=660). As expected, the more inducing variables, the better the fit.

We also report the wall clock time for the experiments (training and evaluation) for $N = 10,000$ and $N = 5,929,413$. All these experiments were ran on a single consumer-grade GPU (Nvidia GTX 1070). On the complete dataset of almost 6 million records, VISH took 41 ± 0.81 seconds on average. A-VFF required 75.61 ± 0.75 seconds and the SVGP method needed approximately 15 minutes to fit and predict. This shows that VISH is roughly two orders of magnitude faster than SVGP. A-VFF comes close to VISH but has to impose additive structure to keep its computational advantage.

3.6 Concluding Remarks

In this chapter we introduced the RKHS associated with dot-product kernels on the hypersphere. Using the Mercer representation of this RKHS we designed an efficient spherical harmonic inducing function for sparse variational Gaussian processes. Our general setup is closely related to VFF, and we inherit several of its advantages such as a considerable speed-up compared to classic sparse GP models as a result of sparsity in the $\mathbf{K}_{\boldsymbol{uu}}$ matrix, and having fixed features $\mathbf{k}_{\boldsymbol{u}\cdot}$ .with a global influence on the approximation. However, by projecting the data onto the hypersphere and using dedicated GP models on this manifold our approach succeeds where other sparse GPs methods fail. First, VISH provides good scaling properties as the dimension of the input space increases. Second, we showed that under some relatively weak hypothesis we are able to select the optimal features to include in the approximation. This is due to the intricate link between "stationary" covariances on the sphere and the Laplace-Beltrami operator. Third, the Mercer representation of the kernel means that the matrices to be inverted at training time are exactly diagonal —resulting in a very cheap-to-compute sparse approximate GP.

In the next chapter, we are going to employ the same RKHS. However, instead of using the eigenfunctions of the kernel operator as the inducing functions, we are going to rely on the reproducing property to construct inducing functions that closely mimic the activation functions in neural networks.

Chapter 4

Deep Neural Networks as Point Estimates for Deep Gaussian Processes

Neural networks (NNs) and Gaussian processes (GPs) are complementary in their strengths and weaknesses. Having a better understanding of their relationship comes with the promise to make each method benefit from the strengths of the other. In this chapter, we establish an equivalence between the forward passes of neural networks and the posterior of deep sparse Gaussian process models. The theory we develop builds upon the RKHS presented in chapter 3 and is based on interpreting activation functions as interdomain inducing features through a rigorous analysis of the interplay between activation functions and kernels. This results in models that can either be seen as neural networks with improved uncertainty prediction or deep Gaussian processes with increased prediction accuracy. In section 4.1 we first give a brief overview of the rich literature on Bayesian treatments of NNs and how this gives rise to (deep) GPs. In section 4.2 we then formally introduce Deep Gaussian processes (DGPs) and highlight the close similarly between NNs and the posterior of sparse DGPs. In section 4.3 we present our *activated* inducing variables for which the resulting GP basis functions match traditional neural network activation, which gives rise to the equivalence between DGPs and NNs. Our claims are supported by experimental results on the UCI repository benchmarks and a series of image classification tasks in section 4.4.

4.1 Related Work

Many different relationships between GPs and NNs have been established over the years. These relationships mainly arise from Bayesian approaches to neural networks. Finding the posterior distribution on neural network weights is challenging, as closed-form expressions do not exist. As a consequence, developing accurate approximations has been a key goal since the earliest

works on Bayesian NNs [MacKay, 1992a]. While investigating single-layer NN posteriors, Neal [1995] noticed that randomly initialised NNs converged to a *Gaussian process* (GP) in the limit of infinite width. Like NNs, GPs represent functions, although they do not do so through weights. Instead, GPs specify function values at observed points, and a kernel which describes how function values at different locations influence each other.

This relationship was of significant practical interest, as the mathematical properties of GPs could **1)** represent highly flexible NNs with *infinite* weights, and **2)** perform Bayesian inference without approximations. This combination was highly successful for providing accurate predictions with reliable uncertainty estimates [Williams and Rasmussen, 1996; Rasmussen, 1997]. To obtain models with various properties, GP analogues were found for infinitely-wide networks with various activation functions [Williams, 1998] including the ReLU [Cho and Saul, 2009]. More recently, Meronen et al. [2020] investigated the relationship in the opposite direction, by deriving activation functions such that the infinite width limit converges to a given GP prior.

Since the growth of modern deep learning, relationships have also been established between infinitely wide *deep* networks and GPs [Matthews et al., 2018; Lee et al., 2018; Yang, 2019]. Given these relationships, one may wonder whether GPs can supersede NNs, particularly given the convenience of Bayesian inference in them. Empirically though, finite NNs outperform their GP analogues [Lee et al., 2018; Garriga-Alonso et al., 2018; Novak et al., 2019] on high-dimensional tasks such as images. **mackay1998introgp** explained this by noting that NNs lose their ability to learn features in the infinite limit, since every GP can be represented as a single-layer NN, with *fixed* features [Mercer, 1909; Rasmussen and Williams, 2006a]. This observation justifies the effort of performing approximate Baysian inference in *finite* deep networks, so both feature learning and uncertainty can be obtained. Renewed effort in Bayesian training procedures has focussed on computationally scalable techniques that take advantage of modern large datasets, and have provided usable uncertainty estimates [e.g., Kingma et al., 2015; Blundell et al., 2015; Gal and Ghahramani, 2016; Louizos and Welling, 2016].

However, questions remain about the accuracy of these approximations. For accurate Bayesian inference, marginal likelihoods are expected to be usable for hyperparameter selection [MacKay, 1992b; 2003]. For most current approximations, there is either no positive or explicitly negative [Blundell et al., 2015] evidence for this holding, although recent approaches do seem to provide usable estimates [Ober and Aitchison, 2020; Immer et al., 2021].

DGPs [Damianou and Lawrence, 2013] provide an alternative approach to deep NNs, which use GP layers instead of weight-based layers, in order to take advantage of the improved uncertainty estimates afforded by having an infinite number of weights. The DGP representation is particularly promising because both early and recent work [Damianou and Lawrence, 2013; Damianou, 2015; Dutordoir et al., 2020b] shows that marginal likelihood estimates are usable for hyperparmeter selection, which indicates accurate inference. However, currently scalability and optimisation issues hinder widespread use.

4.2 Deep Gaussian Processes

GPs are often used as priors over functions in supervised learning, as for Gaussian likelihoods the posterior $f \,|\, \mathcal{D}$ given a dataset $\mathcal{D} = \{x_i \in \mathbb{R}^d, y_i \in \mathbb{R}\}_{i=1}^N$ is tractable (see section 2.1.1). Surprisingly diverse properties of the GP prior can be specified by the kernel [Rasmussen and Williams, 2006b; Duvenaud et al., 2013; Wilson and Adams, 2013; Ginsbourger et al., 2013; van der Wilk et al., 2018], although many problems require the flexible feature learning that deep NNs provide. DGPs [Damianou and Lawrence, 2013] propose to solve this in a fully Bayesian way by composing several layers of simple GP priors,

$$\mathcal{F} = f_L \circ \ldots \circ f_2 \circ f_1 \qquad \text{where} \qquad f_\ell \sim \mathcal{GP}(0, k_\ell). \tag{4.1}$$

Inference in DGPs is challenging as the composition is no longer a GP, leading to an intractable posterior $\mathcal{F} \,|\, \mathcal{D}$. Various approximations have been developed over the years [Damianou and Lawrence, 2013; Hensman and Lawrence, 2014; T. Bui et al., 2016; Havasi et al., 2018]. We follow the variational approach by Salimbeni and Deisenroth [2017] due to its similarity to backpropagation in NNs. They use an approximate posterior consisting of an independent sparse GP for each layer. Each sparse GP is constructed by conditioning the prior on M inducing variables [Titsias, 2009; Hensman et al., 2013; Matthews et al., 2016], commonly function values $\boldsymbol{u}_\ell = \{u_\ell^m = f_\ell(w_\ell^m)\}_{m=1}^M$, and then specifying their marginal distribution as $q(\boldsymbol{u}_\ell) = \mathcal{N}(\boldsymbol{m}_\ell, \mathbf{S}_\ell)$. This results in the approximate posterior process:

$$q(f_\ell(x)) = \mathcal{GP}\Big(\mathbf{k}_{\boldsymbol{u}_\ell}^\top(x)\mathbf{K}_{\boldsymbol{u}_\ell\boldsymbol{u}_\ell}^{-1}\boldsymbol{m}_\ell, \quad k_\ell(x,x') - \mathbf{k}_{\boldsymbol{u}_\ell}^\top(x)\mathbf{K}_{\boldsymbol{u}_\ell\boldsymbol{u}_\ell}^{-1}(\mathbf{K}_{\boldsymbol{u}_\ell\boldsymbol{u}_\ell} + \mathbf{S}_\ell)\mathbf{K}_{\boldsymbol{u}_\ell\boldsymbol{u}_\ell}^{-1}\mathbf{k}_{\boldsymbol{u}_\ell}(x)\Big). \tag{4.2}$$

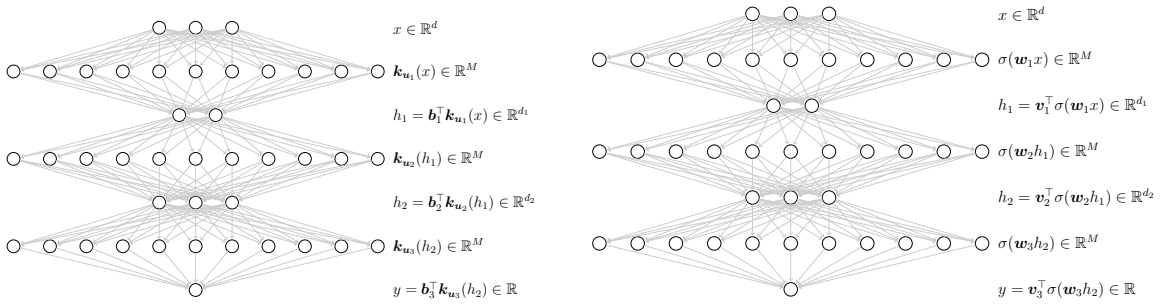
The variational parameters $\boldsymbol{m}_\ell \in \mathbb{R}^M$ and $\mathbf{S}_\ell \in \mathbb{R}^{M\times M}$ are selected by reducing the Kullback-Leibler (KL) divergence from the variational distribution to the true posterior. This is equivalent to maximising the Evidence Lower BOund (ELBO). Taking the evaluations of the GP as $\boldsymbol{h}_{:,\ell} = f_\ell(\boldsymbol{h}_{:,\ell-1})$, the ELBO becomes [Salimbeni and Deisenroth, 2017]:

$$\log p(\boldsymbol{y}) \geq \sum\nolimits_i \mathbb{E}_{q(h_{i,L})}[\log p(y_i \,|\, h_{i,L})] - \sum\nolimits_\ell \mathrm{KL}[q(\boldsymbol{u}_\ell) \,\|\, p(\boldsymbol{u}_\ell)]. \tag{4.3}$$

4.2.1 Connection to Deep Neural Networks

Investigating eq. (4.2), it is worth noting that the posterior $q(f_\ell)$'s variance takes into account the infinite degrees of freedom from the prior; making predictions with an infinite amount of basis functions. The mean, on the contrary, has a finite numbers of parameters and can straightforwardly be re-written as a linear model with M basis functions of the form $\mathbf{k}_{\boldsymbol{u}_\ell}(x) = \mathrm{Cov}(\boldsymbol{u}_\ell, f_\ell(x))$:

$$\mathbb{E}_{f_\ell}[q(f_\ell(x))] = \boldsymbol{b}_\ell^\top \mathbf{k}_{\boldsymbol{u}_\ell}(x) \quad \text{where} \quad \boldsymbol{b}_\ell = \mathbf{K}_{\boldsymbol{u}_\ell\boldsymbol{u}_\ell}^{-1}\boldsymbol{m}_\ell \in \mathbb{R}^M \tag{4.4}$$



(a) Deep Gaussian process posterior (b) Fully-connected deep neural network

Fig. 4.1 Forward pass through a DGP **(a)** and a DNN **b**. By matching basis functions $\mathbf{k}_{\boldsymbol{u}_\ell}(x)$ with $\sigma(\boldsymbol{w}_\ell x)$ and weights $\boldsymbol{b}_\ell$ with $\boldsymbol{v}_\ell$ we obtain an equivalence between both models.

The mean of a layer in DGP is therefore "not that different" from a fully-connected layer in a neural network (NN-FCL), which admits the following structure

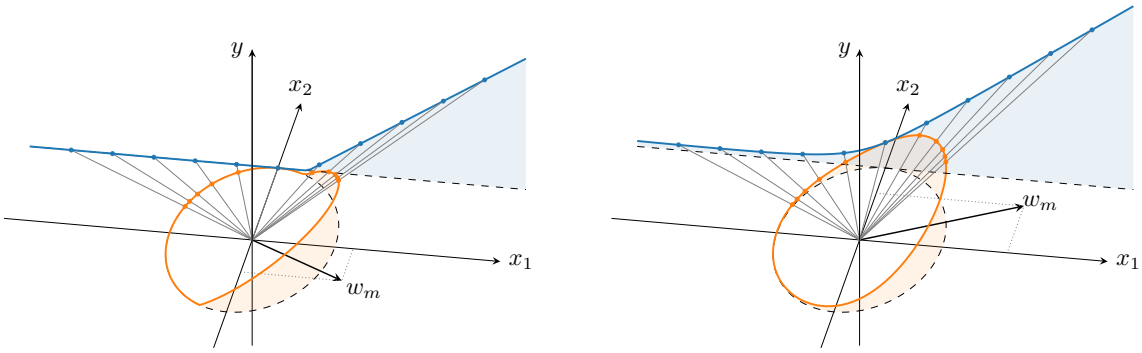
$$\text{NN-FCL}_\ell(x) = \boldsymbol{v}_\ell^\top \sigma(\boldsymbol{w}_\ell\, x), \tag{4.5}$$

where σ is the non-linear activations function (e.g., ReLU), $\boldsymbol{w}_\ell \in \mathbb{R}^{M\times d}$ and $\boldsymbol{v}_\ell \in \mathbb{R}^{M\times d_\ell}$ are the pre-activation and output weights, respectively. Looking at the resemblance between eq. (4.4) and eq. (4.5), if we are able to formulate an inducing variable that makes the covariance $\mathbf{k}_{\boldsymbol{u}_\ell}(x)$ the same as a neural net activation function $\sigma(\boldsymbol{w}_\ell\, x)$, we will have a formal mathematical equivalence between the mean of GP layer in a DGP and a fully-connected neural network layer, that is $\mathbb{E}[q(f_\ell(x))] = \text{NN-FCL}_\ell(x)$. By stacking many of these layers we obtain an equivalence between the forward pass in a DGP and a DNN, as visualised in fig. 4.1.

In the next section we design the interdomain inducing variable for which the covariance will match the neural network activation function. This is the main contributions of this chapter.

4.3 Gaussian Process Layers with Activated Inducing Variables

Building on the exposition of zonal kernel RKHSs (section 3.1) and interdomain inducing variables (section 2.2.1) we can summarise our approach succinctly: we consider DGP models with GP layers that have an Arc Cosine kernel and Variational Fourier Feature style inducing variables $u_m = \langle f, g_m\rangle_{\mathcal{H}}$ [Hensman et al., 2018]. In section 4.3.1, we then design inducing functions g_m that have the same shape as neural network activation functions. Thanks to the reproducing property of the RKHS this yields basis functions $\mathbf{k}_{\boldsymbol{u}_\ell \cdot}$ for the GP layers that correspond to activation functions (section 4.3.2), and thus to a DGP whose forward pass can be interpreted as a classic feed forward neural network. Section 4.3.3 covers the mathematical intricacies associated with the construction described above.



(a) ReLU: $\sigma(t) = \max(0, t)$ (b) Softplus: $\sigma(t) = \log(1 + \exp(3\,t))$

Fig. 4.2 Activated inducing function $g_m(x) = ||x||\,||w_m||\,\sigma\Big(w_m^\top x \,/\, ||w_m||\,||x||\Big)$ where σ correspond to the ReLU **(a)** and Softplus **(b)**. Although the input domain is $\mathbb{R}^2$ we only plot the value of the function on the unit circle $\mathbb{S}^1$ (orange), and on the subspace that has an offset of 1 in the x_2 direction (blue). The linear radial component of g_m creates a one-to-one mapping between the blue curve and the upper half of the orange one.

4.3.1 Activated Inducing Functions

Preliminary: the bias term. We consider the RKHS of the first-order Arc Cosine kernel as discussed in section 3.3.1, which consists of functions of the form $f(x) = ||x||g(\frac{x}{||x||})$. As a result, all function in $\mathcal{H}$ are equal to zero at the origin, i.e. $\forall f \in \mathcal{H} : f(0) = 0$, which is a very restrictive modelling assumption. 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.

Design of the inducing function. The inducing functions g_m 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 that is not necessarily zonal. Since this property will be important later on, we favour the following alternative definition that enforces zonality:

$$g_m : \mathbb{R}^d \to \mathbb{R}, \qquad x \mapsto ||x||\,||w_m||\,\sigma\left(\frac{w_m^\top x}{||w_m||\,||x||}\right), \tag{4.6}$$

with $w_m \in \mathbb{R}^d$ a parameter, and $\sigma : [-1, 1] \to \mathbb{R}$ the function that determines the value of g_m on the unit hypersphere. In fig. 4.2, we show that choosing σ 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 $w_m \in \mathbb{R}^d$ 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. (4.6) is particularly convenient when it comes to representing g_m 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 eq. (B.1)) and to obtain

$$\begin{aligned} g_m(x) = ||x||\,||w_m||\sum_{n=0}^{\infty}\sum_{j=1}^{N_n^d}\sigma_n\phi_{n,j}\left(\frac{w_m}{||w_m||}\right)\phi_{n,j}\left(\frac{x}{||x||}\right), \\ \text{where} \quad \sigma_n = \frac{\omega_d}{C_n^{(\alpha)}(1)}\int_{-1}^{1}\sigma(t)\,C_n^{(\alpha)}(t)\,(1-t^2)^{\frac{d-3}{2}}\,\mathrm{d}t. \end{aligned} \tag{4.7}$$

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

4.3.2 Activated Interdomain Inducing Variables

We define our *activated* interdomain inducing variables as

$$u_m = \langle f, g_m\rangle_{\mathcal{H}}. \tag{4.8}$$

Since the GP samples do not belong to the RKHS there are mathematical subtleties associated with such a definition, which are detailed in section 4.3.3. 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 $\mathbf{K}_{\boldsymbol{uu}}$, is given by

$$\begin{aligned} \mathrm{Cov}(u_m, u_{m'}) &= \mathrm{Cov}\Big(\langle f, g_m\rangle_{\mathcal{H}}, \langle f, g_{m'}\rangle_{\mathcal{H}}\Big) && \text{Definition } u_m \\ &= \langle g_m, g_{m'}\rangle_{\mathcal{H}} && \text{Definition RKHS} \\ &= \sum_{n=0}^{\infty}\sum_{j=1}^{N_n^d}\frac{\sigma_n\phi_{n,j}(w_m/||w_m||)\sigma_n\phi_{n,j}(w_{m'}/||w_{m'}||)}{\lambda_n} && \text{Fourier coefficients from eq. (4.7)} \\ &= \sum_{n=0}^{\infty}\frac{\sigma_n^2}{\lambda_n}\,\frac{n+\alpha}{\alpha}\,C_n^{(\alpha)}\left(\frac{w_m^\top w_{m'}}{||w_m||\,||w_{m'}||}\right) && \text{Addition theorem 5.} \end{aligned} \tag{4.9}$$

Secondly, the covariance between the GP and u_m, which determines $[k_u(x)]_m$, is given by:

$$\mathrm{Cov}(u_m, f(x)) = \langle k(x,\cdot), g_m\rangle_{\mathcal{H}} = g_m(x) \tag{4.10}$$

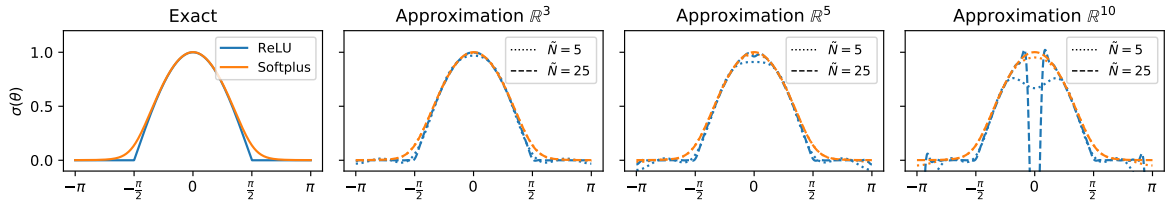


Fig. 4.3 The exact ReLU and Softplus activation function and its approximation for different truncation levels and dimensions. These functions correspond to the orange function in fig. 4.2 plotted on a line rather than on the circle.

as a result of the reproducing property of the RKHS. It becomes clear that this procedure gives rise to basis functions $\mathbf{k}_{\boldsymbol{u}}$. that are equal to our inducing functions g_m. By construction, these inducing functions match neural network activation functions in the data plane, as shown in fig. 4.2. Returning to eq. (4.4), we see that 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).

4.3.3 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. (4.8), 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 eq. (2.30) implies that we have direct access to the Karhunen–Loève representation of the GP:

$$f(x) = \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} \xi_{n,j} \sqrt{\lambda_n} ||x|| \phi_{n,j}\left(\frac{x}{||x||}\right), \quad \text{where the } \xi_{n,j} \text{ are i.i.d. } \mathcal{N}(0,1). \tag{4.11}$$

Using this expression to compute the RKHS norm of a GP sample f 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 $\langle f, g \rangle_{\mathcal{H}}$ should be manipulated with care. By definition, the RKHS inner product is a series that converges if computed for any two elements $g, h \in \mathcal{H}$. 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 is too slow to make it an element of $\mathcal{H}$, if the Fourier coefficients of g decay quickly enough for the series $\sum_{n,j} \xi_{n,j} g_{n,j} / \sqrt{\lambda_n}$ to converge then $\langle f, g \rangle_{\mathcal{H}}$ is well defined.

The above reasoning indicates that, for a given kernel k, some activation functions g_m will result in inducing variables $u_m = \langle f, g_m \rangle_{\mathcal{H}}$ 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. We notice this behaviour in the bottom plots of fig. 4.4, where the norm $||g_m||$ in the case of the ReLU (left) keep growing with N, whereas the norm for the softplus converges. This will have implications on the GP fit, as later discussed.

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 $\tilde{g}_m$ 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. Figure 4.3 shows how the true and truncated activation functions for the ReLU and Softplus compare. These correspond to the orange functions in fig. 4.2, 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.

Inexpressive inducing variables through truncation (fig. 4.4) The main concern with this truncation approach, however, comes from elsewhere: the larger $\tilde{N}$ is, the closer $\tilde{g}_m$ is to g_m, but the larger $||\tilde{g}_m||_{\mathcal{H}}$ 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. Figure 4.4 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.

Inexpressive inducing variables through spectra mismatch (fig. 4.5) 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 kernel because we can replace the shape function in eq. (3.25) by any stationary kernel, and our approach would still hold. For example, we could transform the Matérn-5/2 to a zonal kernel as follows

$$k(x, x') = ||x||||x'||\kappa_{\text{mat-5/2}}\Big(\frac{x^\top x'}{||x||||x'||}\Big), \text{ with } \kappa_{\text{mat-5/2}}(t) = \gamma^2\left(1 + \frac{\sqrt{5}t}{\rho} + \frac{5t^2}{3\rho^2}\right)\exp\left(-\frac{\sqrt{5}t}{\rho}\right),$$

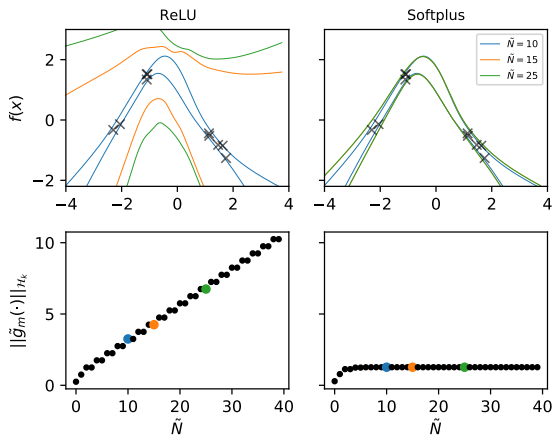


Fig. 4.4 **Top:** Predictive variance an SVGP fit on a synthetic dataset using our ReLU (left) and Softplus (right) inducing variables for $\tilde{N} = 10, 15$ and 25. **Bottom:** the norm of the inducing function $\tilde{g}_m$ as a function of $\tilde{N}$.

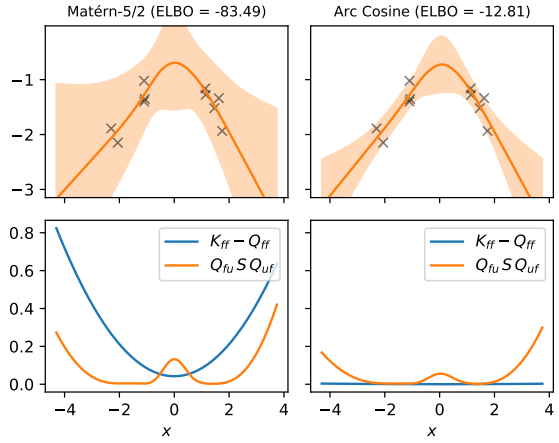


Fig. 4.5 **Top:** Predictive mean and variance for an SVGP model using our Softplus inducing variables for the Matérn-5/2 (left) and Arc Cosine (right) kernel. **Bottom:** The two terms that constitute the predictive variance.

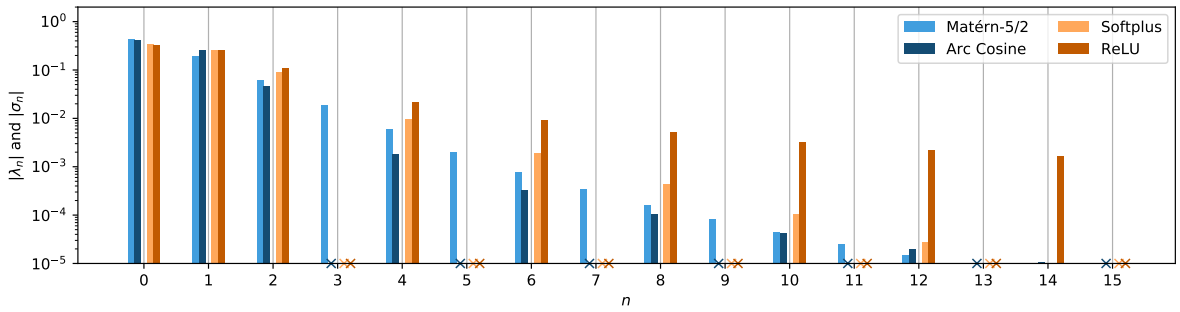


Fig. 4.6 Spectra of Arc Cosine and Matérn-5/2 (blue), and ReLU and Softplus (orange) for different levels.

γ^2 is the variance and ρ the lengthscale. However, in fig. 4.5 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.

In the bottom panel in fig. 4.5 we see that for the Matérn-5/2, the Nyström approximation $\mathbf{Q_{ff}} = \mathbf{K_{fu}}\mathbf{K_{uu}^{-1}}\mathbf{K_{fu}^{\top}}$ is unable to explain the prior $\mathbf{K_{ff}}$, imposed by the kernel. This leads to the overestimation of the predictive variance in the top plot. The reason for this problem is the mismatch between the eigenvalues λ_n (**??**) of the Matérn and the Fourier coefficients σ_n (eq. (4.7)) of the Softplus inducing function. As shown in fig. 4.6, the Matérn kernel has a full spectrum (i.e., $\lambda_n \neq 0,\ \forall n \in \mathbb{N}$), whereas the coefficients for the Softplus σ_n are zero at levels $n = 3, 5, 7, \cdots$. We are thus trying to approximate our prior kernel, containing all levels, by

a set of functions that is missing many. This problem does not occur for the combination of Softplus (or ReLU) inducing functions and the Arc Cosine kernel (right-hand side) because their spectral decomposition match. In other words, the Arc Cosine kernel has zero coefficients for the same levels as our activated inducing functions.

Taking the above two cases into consideration which lead to inexpressive inducing variables, in the upcoming experiments we use the Arc Cosine kernel and inducing variables obtained with the Softplus activation (section 4.3.1) with $\tilde{N} = 20$.

4.4 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.

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.

4.4.1 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 $u_m = f(z_m)$, 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.

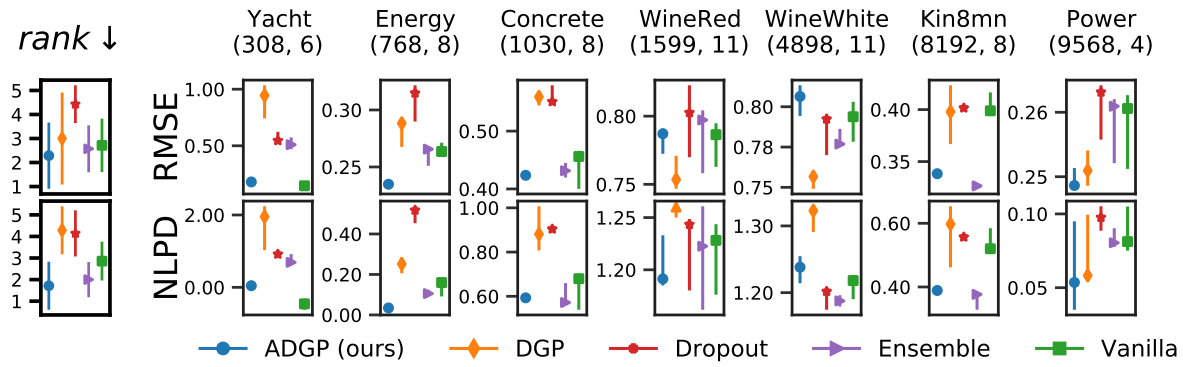


Fig. 4.7 UCI: Root Mean Squared Error (RMSE) and Negative Log Predictive Density (NLPD) with 25% and 75% quantile error bars based on 5 splits. Dataset size and dimension given in parentheses.

In fig. 4.7 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.

4.4.2 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. As previously, the ADGP model is initialised to the solution of the Vanilla model, and training is then continued using the ELBO. In fig. 4.8 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.

4.5 Conclusion

In this chapter, we establish a connection between fully-connected neural networks and the posterior of deep sparse Gaussian processes. We use a specific flavour of interdomain inducing

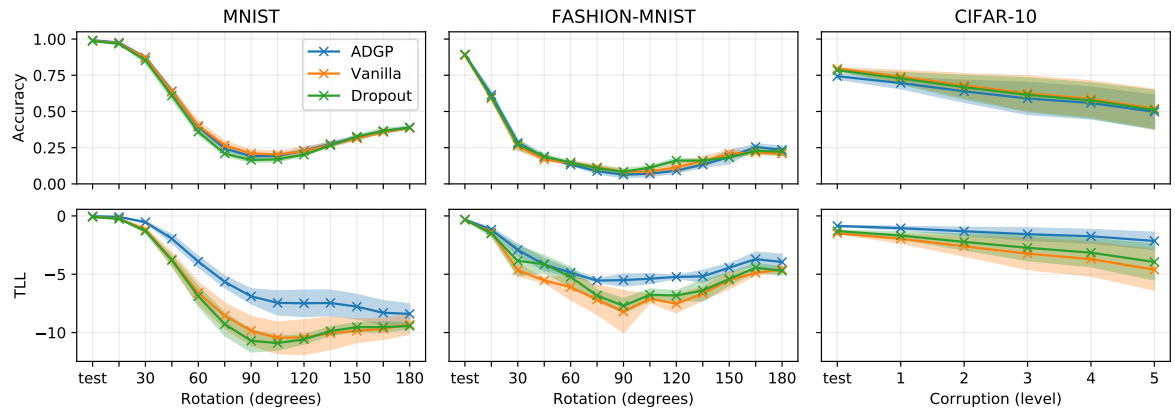


Fig. 4.8 Results on the rotated MNIST, FASHION-MNIST and corrupted CIFAR-10, showing the mean and std. dev. of the accuracy **(top)**, and test log-likelihood (TLL) **(bottom)**.

variables based on the RKHS inner product to obtain basis functions for the SVGP that match activation functions from the neural network literature. By composing such SVGPs together, we obtain a DGP for which a forward pass through the mean of each layer is equivalent to a forward pass in a DNN. We also address important mathematical subtleties to ensure the validity of the approach and to gain insights on how to choose the activation function.

Chapter 5

Future Research Agenda

5.1 Introduction

To date, my research has focussed on using Gaussian processes (GPs) as building blocks for deep models, similar to neural network layers. In this vein, I have worked on (1) deep convolutional Gaussian processes that mimic the convolutional and layered architectures encountered in many contempory deep learning models [Dutordoir et al., 2020b], (2) conditional density estimation models which are similar to Variational Auto Encoders [Dutordoir et al., 2018; Salimbeni et al., 2019], and (3) more recently, deep Gaussian processes that have basis functions that are similar to neural network activation functions [Dutordoir et al., 2021a] (discussed at length in chapter 4). I have packaged these advances in a software toolbox, called GPflux [Dutordoir et al., 2021b], which provides these GP layers through a neural network like interface. The main advantage of this approach is that straightforward application of Bayesian principles leads to sensible results, which is not necissarily the case for normal neural networks [Wenzel et al., 2020]. The current state of research into using Gaussian processes as layers is that classification performance starting to be on-par with deep learning on simple datasets [Dutordoir et al., 2018], but with good uncertainty quantification and automatic selection of hyperparameters working 'out of the box'.

The primary focus of my previous research can be considered as 'curve-fitting', i.e. finding the best function approximator when given examples of inputs and corresonponding outputs. In this setting we can consider two regimes: (1) the function is very complex but there is an abundance of high-quality and cheap data to learn the mapping, and (2) the function is of a simpler nature but the underlying data is limited, noisy and/or very expensive to acquire. As evidenced by many recent advances in domains such as natural language and vision, where indeed there is a large availablility of high-quality data, the first regime is a setting where deep learning thrives. The natural inductive bias of deep neural networks (DNNs) in combination with the low computational cost of training them on massive datasets has proven to be very

effective. Arguably, performing (approximate) Bayesian inference in this regime will contribute little to the end result, and the computational cost associated with it makes it very cumbersome.

The second regime, in which data is noisy, requires a completely different modelling paradigm. On the one hand, the noisy data forces the model to be uncertain about the signal that can be extracted from the given examples. On the other hand, the simpler nature of the function allows an expert to encode prior information about the problem at hand. This enables faster generalisation from little data, which can be necessary given the high cost of acquiring more. This is a setting where probabilistic Bayesian models thrive.

However, to fully valorise Bayesian models one needs to consider them as part of a larger 'decision-making' system. In these systems probabilistic models can be used to determine optimal actions to take in order to achieve a certain outcome.

In the next chapter of my PhD, I want to evolve from purely 'curve-fitting' applications to focusing more on decision-making systems that *use* probabilistic models to drive their decisions. In particular, I want to focus on probabilistic models that take into account the geometry of the problem at hand. This will allow encorporating more accurately physical properties and constraints of the system under study. In the next section we briefly discuss Markov Decision Processes (MDPs), a formal mathematical framework to decision making. We pay attention to the model based approach. InSection we can define probabilistic models in non-Euclidian spaces. on . We finish with suggesting concrete research topics and a real-world application.

5.2 Markov Decision Process

A discrete-time Markov decision Process (MDP) is a stochastic control process. It provides a mathematical framework for decision making in situations where outcomes are partly random and partly under the control of a decision maker [CITE TODO]. A MDP is characterised by a 4-tuple, consiting of

1. A state space $\mathcal{S}$,
2. An action space $\mathcal{A}$,
3. Reward density $p(r \mid a, s)$: the immediate reward obtained after executing an action a in a given state s,
4. Transition density $p(s' \mid a, s)$: transitioning from state s to state s', due to action a.

It should be noted that this is a very broad definition, yet there are still many variations to it. For instance, one can consider deterministic reward and transition functions, continuous-time and partially observed MDPs.

The objective of an MDP is to find a 'good' (probabilistic) policy function $\pi : \mathcal{S} \to \mathcal{X}$. This is a mapping that specifies which action a decision maker should take in a given state s.

The goal is to find a sequence of actions, starting from an initial state s_0, that maximses the cumulative reward a decision maker obtains by obeying the policy π. This is given by the value function

$$V^{(\pi)}(s_0) = \mathbb{E}\left[\sum_t r_t\right] \tag{5.1}$$

where the expectation is over $r_t \sim p(r_t \mid s_t, a_t)$, $s_{t+1} \sim p(s_{t+1} \mid s_t, a_t)$ and $a_t \sim \pi(s_t)$.

The quality of a policy π is given by the *regret*, which measures the value function $V^{(\pi)}$ relative to the optimal value function V^* obtained by the best possible policy π^*

$$R^{(\pi)}(s_0) = V^*(s_0) - V^{(\pi)}(s_0). \tag{5.2}$$

5.2.1 A Model-based Approach

There are many different ways to deduct a good—or even an optimal—policy for a given MDP, but the exact algorithm will heavily depend on the precise formulation of the problem. Here, we consider the case that the transition and reward function are unknown, and also very expensive to evaluate. In this setting, any efficient algorithm will need to consider whether to take explorative actions that might lead to better rewards, or take advantage of actions that are known to be good but will not improve the current solution. This dilema is known as the exploitation-exploration trade-off. The fact that the functions are expensive to evaluate makes it worthwile to spend resources (e.g., time and compute) on deciding where to evaluate them next.

A strategy to address this MDP formulation is using a *model-based* approach. Broadly speaking, this approach consists of a model that learns the unknown transitions and/or rewards from observed data in a supervised learning way. Subsequently, a policy is derived using the learned model rather than using the true reward and transition functions.

Example We consider Bayesian optimisation (BO) [CITE] as an illustration of these concepts. BO is a sequential search algorithm for finding a global minimiser of an unknown objective function f defined over a domain of interest $\mathcal{X}$

$$x_* = \operatorname*{argmin}_{x \in \mathcal{X}} f(x). \tag{5.3}$$

The function f is not observed directly. Instead, at each step the algorithm selects a query point $x \in \mathcal{X}$ at which to evaluate f and obtains $y(x)$: a function evaluation corrupted by noise $y(x) \sim p(y(x) \mid f(x))$.

BO can be framed as a MDP if one considers the cardinality of $\mathcal{S}$ to be 1, and the actions space to cover the function's input domain $\mathcal{A} = \mathcal{X}$. As there is only a single state the transition

functions trivially reduces to the identity function. The reward associated to an action x[1] is given by the negative function observation. This construction leads to an unknown and probabilistic reward function due to the unknown objective function and the noisy observations. Collecting the sequence of actions (i.e. query points) $\{x_t\}$, the performance of an algoritm selecting the query points can be measured using a tailered notion of reget

$$R = \sum_t f(x_t) - f(x_*). \tag{5.4}$$

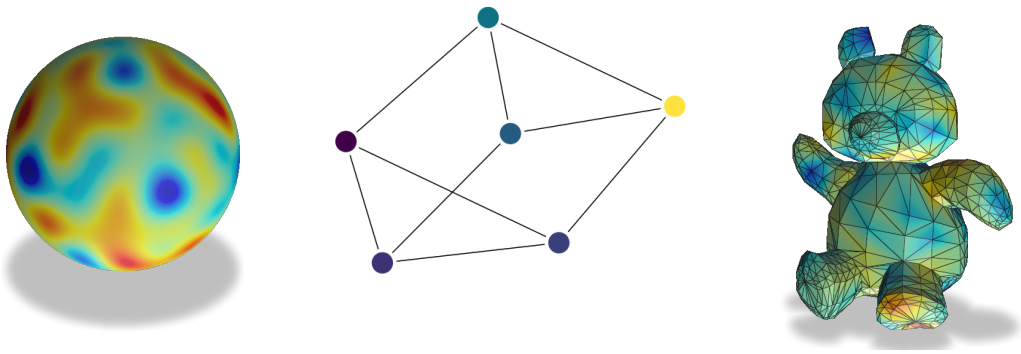
A model-based approach to this problem employs a surrogate model g which learns the unknown objective function f using past observations. The surrogate model is used to decide on the next query point, through what is known as an *acquisition rule* [CITE]. The exploitation-exploration trade-off highlights the importance of the surrogate model's faithful represention of uncerainty about the objective. It is a key ingredient to an effective algorithm and is necessary to obtain a low regret [CITE]. Most model-based approaches employ Gaussian processes (GPs) as their surrogate model: $g \sim \mathcal{GP}$ [CITE].

To follow a model-based approach in Bayesian optimisation, as well as in other kinds of MDPs, it is required to define a GP on the domain of interest. For convenience, this domain is typically chosen to be the Euclidian space. However, many problems possess important non–Euclidean geometric structure, which makes it much more effective and convenient to directly specify the problem on the non-Euclidean domain, like the torus or the sphere, but also a graph, a rotation group, or the space of positive-definite matrices. Consider for example optimising the joint postures of a robot, which are defined on the torus $\mathcal{T}^d = \mathcal{S}^1 \times \mathcal{S}^1 \ldots \times \mathcal{S}^1$. To accomodate for this, one must be able to specify a GP prior, or equivalently define a kernel, on this space of interest.

5.3 Gaussian Processes as Surrogate Models in non-Euclidian manifolds

Stationary kernels are ubiquitous in Gaussian processes when the input space is Euclidean. Their popularity is due to their general useability and the ease of understanding which prior information they encode. Recent work of Borovitskiy et al. [2020] has extended the definition of stationary kernels to Riemannian manifolds. This enables the definition of symmetric and positive-definite kernels, which characterise GPs, that take into account the geometry of the space. In the case of a graph, for example, two nodes can be close to each other in the Euclidean space yet far apart in the term of number of edges them separates them. A classical statioary kernel is not able to capture this information, whereas a geometric aware kernel can.

[1] To adhere to the common notation in BO we use x, rather than a, to represent the action (in our case the next query point).



(a) Sphere (b) Graph (c) Mesh

Fig. 5.1 Matérn GP samples on non-Euclidian spaces

The construction in Borovitskiy et al. [2020] relies on the spectral theory of the Laplace-Beltrami operator and defines the kernel through its Mercer decomposition (eq. (2.30))

$$k(x, x) = \sum_i S(\sqrt{\lambda_i})\, \phi_i(x)\, \phi_i(x') \tag{5.5}$$

where S is the power spectrum of the kernel and $\{\lambda_i, \phi_i\}$ are the eigenvalues and eigenfunctions of the Laplace-Beltrami operator. chapter 3 discusses a special case of this construction for zonal kernels defined on the hypersphere where the eigenfunctions are given by the spherical harmonics.

The elegance of eq. (5.5) is that it defines the kernel through the eigenfunctions of the Laplace-Beltrami operator. These eigenfunctions are known analytically for certain well-studied manifolds, such as the sphere and the torus. However, in practice, working with more complex manifolds involves descretizing the manifold of which the mesh and undirected weighted graph are special cases. This simple framework allows us to unify these spaces and allows us to develop apply to all of them.

Given the brief background on model-based approaches to MDP and Gaussian processes defined on non-Euclidian spaces we now suggest the following projects

5.4 Research Agenda

The overarching theme of my proposed research agenda is: "How can we make Gaussian processes an effective tool for decision-making tasks on non-Euclidean spaces?". The goal would be that these tools are more widely used outside our 'small' GP community by scientist and engineers who deal with these problems regularily. One could envision these tools to be particularly useful in domains such as mechanical engineering where finite element meshes, environmental modelling which is often done on the sphere, and bio-medical applications on organ-shaped manifolds. To achieve this goal we need at least the following components:

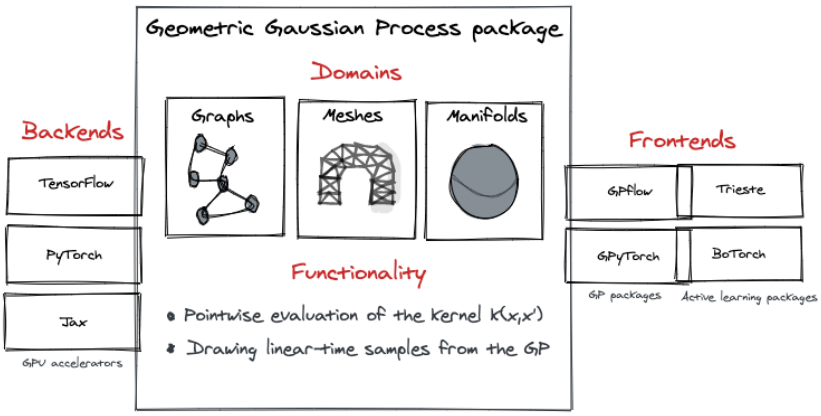


Fig. 5.2 Overview of Geometric Kernel package

5.4.1 Adopotion: Geometric Kernel Package

Open-source, high-quality software is a key katalysator for good research: it allows one to quickly prototype ideas on top of strong foundations, but—more importantly—it makes it possible to package mature research work to make it available to the wider community. The first objective is to develop a Python package which provides implementations of geometric kernels on analytic manifolds and. The package will also be opinionated on the best way to handle these objects for downstream tasks. However, to not hinder adoption by the wider community the package will be backend agnostic, which will allow for downstream use TensorFlow package (e.g., GPflow) and PyTorch packages (e.g., GPytorch). Figure 5.2 gives an overview of the package. Development has started at https://github.com/GPflow/GeometricKernels.

5.4.2 Efficiency: Sparse Manifold Gaussian processes

The cubic computational complexity of GPs in Euclidian spaces limits their use in many real-world applications. As explained in chapter 2, a widely adopted solution to this problem is to employ a smaller set of inducing points which summarise the GP at a set of optimised input locations. Not surprisingly, the same cubic computational complexity is present in the non-Euclidian counterpart of GPs. It is, unfortunately, not straightforward to extend the inducing point solution to these domains. For example, it is not clear how to define an inducing variable on a graph or on a mesh, and how one would freely optimise their location to minimse the KL divegence between true and approximate posterior—as is the case with traditional inducing point inputs. The interdomain and eigenfeatures inducing variable approach, introduced in chapters 2 and 3, seems a promising route as the resulting basisfunctions are fixed, and by construction optimally placed for a given geometry. The objective of this research project is to develop a scalable method for GPs on manifolds, akin to the methods that one has at their disposable in the Euclidian case.

5.4.3 Application: Sequential decision-making on non-Euclidian spaces

To convince a wider audience to use these methods, it is important to show their practical usefulness on a real-world use-case.

Take into account the case of moving the search operation. Calculate expecations of costs associated to search missions.

To illustrate this it woul Bayesian search: airplane crash -> gravitational waves

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 (NeurIPS)*.

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)*.

Viacheslav Borovitskiy, Alexander Terenin, Peter Mostowsky, and Marc P. Deisenroth (2020). "Matern Gaussian processes on Riemannian manifolds". In: *Advances in Neural Information Processing Systems 33 (NeurIPS)*.

Thang Bui, Daniel Hernández-Lobato, José Miguel Hernández-Lobato, Yingzhen Li, and Richard E. Turner (2016). "Deep Gaussian Processes for Regression using Approximate Expectation Propagation". In: *Proceedings of the 33rd 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 (NeurIPS)*.

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

Andreas Damianou (2015). "Deep Gaussian Processes and Variational Propagation of Uncertainty". PhD thesis. University of Sheffield.

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*.

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, James Lloyd, Roger Grosse, Joshua Tenenbaum, and Ghahramani Zoubin (2013). "Structure discovery in nonparametric regression through compositional kernel search". In: *International Conference on Machine Learning*. PMLR.

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*.

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)*.

David Ginsbourger, Nicolas Durrande, and Olivier Roustant (2013). "Kernels and designs for modelling invariant functions: from group invariance to additivity". In: *mODa 10–Advances in Model-Oriented Design and Analysis*.

Marton Havasi, José Miguel Hernández-Lobato, and Juan José Murillo-Fuentes (2018). "Inference in deep gaussian processes using stochastic gradient hamiltonian monte carlo". In: *arXiv preprint arXiv:1806.05490*.

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)*.

Alexander Immer, Matthias Bauer, Vincent Fortuin, Gunnar Rätsch, and Mohammad Emtiyaz Khan (2021). *Scalable Marginal Likelihood Estimation for Model Selection in Deep Learning*.

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 (NeurIPS)*.

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

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 (NeurIPS)*.

Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S. Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein (2018). "Deep neural networks as Gaussian processes". In: *6th International Conference on Learning Representation (ICLR)*.

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*.

Christos Louizos and Max Welling (2016). "Structured and efficient variational deep learning with matrix gaussian posteriors". In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*.

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 (NeurIPS)*.

David J. C. MacKay (2003). *Information Theory, Inference and Learning Algorithms*. Cambridge University Press.

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)*.

James Mercer (1909). "Functions of positive and negative type, and their connection the theory of integral equations". In: *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*.

Lassi Meronen, Christabella Irwanto, and Arno Solin (2020). "Stationary Activations for Uncertainty Calibration in Deep Learning". In: *Advances in Neural Information Processing Systems 33 (NeurIPS)*.

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 (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)*.

Sebastian W. Ober and Laurence Aitchison (2020). *Global inducing point variational posteriors for Bayesian neural networks and deep Gaussian processes*.

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*.

Carl E. Rasmussen (1997). "Evaluation of Gaussian processes and other methods for non-linear regression". PhD thesis. University of Toronto, Canada.

Carl E. Rasmussen and Christopher K. I. Williams (2006a). In: *Gaussian Processes for Machine Learning*. Chap. Regression (§2.1 & §2.2).

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

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

Hugh Salimbeni, Vincent Dutordoir, James Hensman, and Marc P. Deisenroth (2019). "Deep Gaussian Processes with Importance-Weighted Variational Inference". In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*.

Edward Snelson and Zoubin Ghahramani (2005). "Sparse Gaussian Processes using Pseudo-inputs". In: *Advances in Neural Information Processing Systems 18 (NeurIPS)*.

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*.

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

Florian Wenzel, Kevin Roth, Bastiaan S Veeling, Jakub Świątkowski, Linh Tran, Stephan Mandt, Jasper Snoek, Tim Salimans, Rodolphe Jenatton, and Sebastian Nowozin (2020). "How good is the bayes posterior in deep neural networks really?" In: *arXiv preprint arXiv:2002.02405*.

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

Christopher K. I. Williams and Carl E. Rasmussen (1996). "Gaussian processes for regression". In:

Andrew Wilson and Ryan Adams (2013). "Gaussian process kernels for pattern discovery and extrapolation". In: *International conference on machine learning*. PMLR.

Greg Yang (2019). "Wide feedforward or recurrent neural networks of any architecture are Gaussian processes". In: *Advances in Neural Information Processing Systems 32 (NeurIPS)*.

Appendix A

Spherical Harmonics on Hyperspheres

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

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

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

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

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*

$$\mathcal{H}_n^d = \{p \in \mathcal{P}_n^d : \Delta p = 0 \text{ and } p : \mathbb{S}^{d-1} \to \mathbb{R}\}. \tag{A.3}$$

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

$$dim(\mathcal{H}_n^d) = \frac{2n+d-2}{n}\binom{n+d-3}{d-1} := N_n^d. \tag{A.4}$$

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

$$\mathcal{H}_n^d = \text{span}\Big(\phi_{n,1}, \ldots, \phi_{n,N_n^d}\Big), \quad \text{and} \quad \langle \phi_{n,j}, \phi_{n',j'} \rangle_{L_2(\mathbb{S}^{d-1})} = \delta_{nn'}\delta_{jj'}. \tag{A.5}$$

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

$$f = \sum_{n=0}^{\infty}\sum_{j=1}^{N_n^d} \widehat{f}_{n,j}\phi_{n,j}, \quad \text{with} \quad \widehat{f}_{n,j} = \langle f, \phi_{n,j} \rangle_{L_2(\mathbb{S}^{d-1})}, \tag{A.6}$$

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*

$$\sum_{j=1}^{N_n^d} \phi_{n,j}(x)\phi_{n,j}(x') = \frac{n+\alpha}{\alpha} C_n^{(\alpha)}(x^\top x') \quad \textit{with} \quad \alpha = \frac{d-2}{2}. \tag{A.7}$$

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* $\eta \in \mathbb{S}^{d-1}$

$$\frac{1}{\Omega_{d-1}} \int_{\mathbb{S}^{d-1}} f(\eta^\top x)\, \phi_{n,j}(x)\, \mathrm{d}\omega(x) = \lambda_n\, \phi_{n,j}(\eta), \tag{A.8}$$

where λ_n *is a constant defined by*

$$\lambda_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{-1}^{1} f(t)\, C_n^{(\alpha)}(t)\, (1-t^2)^{\frac{d-3}{2}}\, \mathrm{d}t, \tag{A.9}$$

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

A.1 Algorithm: 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* $\{\eta_1, \dots, \eta_{N_n^d}\} \in \mathbb{S}^{d-1}$ *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* $\{\eta_1, \dots, \eta_M\} \in \mathbb{S}^{d-1}$ *is called a fundamental system of degree* n *consisting of* M *points on the sphere if*

$$det \begin{bmatrix} C_n^{(\alpha)}(1) & \dots & C_n^{(\alpha)}(\eta_1^\top \eta_M) \\ \vdots & & \vdots \\ C_n^{(\alpha)}(\eta_M^\top \eta_1) & \dots & C_n^{(\alpha)}(1) \end{bmatrix} > 0. \tag{A.10}$$

Finding a basis for $\mathcal{H}_n^d$ is thus equivalent to finding a set of N_n^d points that satisfy eq. (A.10). 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. (A.10). 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

$$\mathbf{M}(\boldsymbol{\eta}, \eta_*) = \left[\begin{array}{c|c} C_n^{\alpha}(\boldsymbol{\eta}\boldsymbol{\eta}^\top) \in \mathbb{R}^{M \times M} & C_n^{(\alpha)}(\boldsymbol{\eta}\eta_*^\top) \in \mathbb{R}^{M \times 1} \\ \hline C_n^{(\alpha)}(\boldsymbol{\eta}\eta_*^\top)^\top \in \mathbb{R}^{1 \times M} & C_n^{(\alpha)}(1) \in \mathbb{R} \end{array}\right], \tag{A.11}$$

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

$$\eta = \underset{\eta_* \in \mathbb{S}^{d-1}}{\operatorname{argmax}} \ \det(\mathbf{M}(\boldsymbol{\eta}, \eta_*)), \tag{A.12}$$

in order to satisfy the condition in eq. (A.10). 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

$$\eta = \underset{\eta_* \in \mathbb{R}^d}{\operatorname{argmin}} \ C_n^{(\alpha)}\left(\frac{\eta_*}{||\eta_*||}\boldsymbol{\eta}^\top\right)\left[C_n^{(\alpha)}(\boldsymbol{\eta}\boldsymbol{\eta}^\top)\right]^{-1} C_n^{(\alpha)}\left(\boldsymbol{\eta}\frac{\eta_*^\top}{||\eta_*||}\right). \tag{A.13}$$

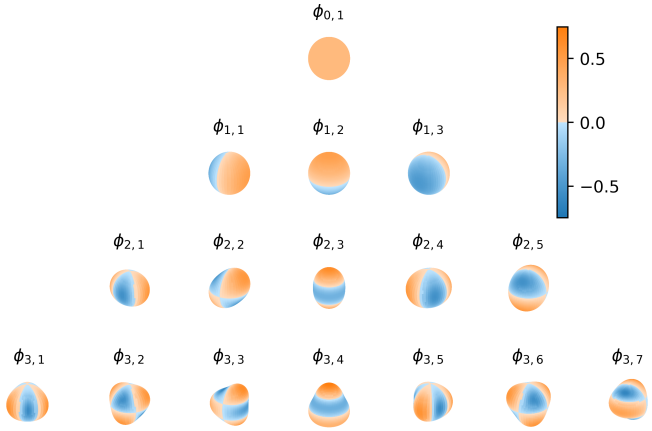


Fig. A.1 Spherical Harmonics

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. (A.13)`

  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*

$$\phi_{n,j}(x) = \sum_{i=1}^{N_n^d} \mathbf{L}_{j,i}\, C_n^{(\alpha)}(\eta_i^\top x) \tag{A.14}$$

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

Appendix B

Analytic computation of eigenvalues for zonal functions

The eigenvalues of a zonal function are given by the one-dimensional integral:

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

where $C_n^{(\alpha)}(\cdot)$ is the Gegenbauer polynomial of degree n with $\alpha = \frac{d-2}{2}$ and $\omega_d = \Omega_{d-2}/\Omega_{d-1}$ denotes the surface area of $\mathbb{S}^{d-1}$ (see **??** for analytical expressions of these quantities). The shape function $s(t)$ determines whether this integral can be computed in closed-form. In the next sections we derive analytical expressions for the eigenvalues of the Arc Cosine kernel and ReLU activation function in the case the d is odd. For d even, other kernels (e.g., Matérn) or activation functions (e.g., Softplus, Swish, etc.) we rely on numerical integration (e.g., Gaussian quadrature) to obtain these coefficients. We will show that both approaches lead to highly similar results.

B.1 Arc Cosine kernel

The shape function of the first-order Arc Cosine kernel [Cho and Saul, 2009] is given by:

$$s : [0, \pi] \to \mathbb{R}, \quad s : x \mapsto \sin x + (\pi - x)\cos x, \tag{B.2}$$

where we expressed the shape function as a function of the angle between the two inputs, rather than the cosine of the angle. For notational simplicity, we also omitted the factor $1/\pi$.

Using a change of variables we rewrite eq. (B.1)

$$\lambda_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{0}^{\pi} s(x)\, C_n^{(\alpha)}(\cos x) \sin^{d-2} x\, \mathrm{d}x, \tag{B.3}$$

Substituting $C_n^{(\alpha)}(\cos x)$ by its polynomial expansion (**??**), it becomes evident that we need a general solution of the integral for n, $m \in \mathbb{N}$

$$\int_0^\pi [\sin(x) + (\pi - x)\cos(x)] \cos^n(x) \sin^m(x) \mathrm{d}x. \tag{B.4}$$

The first term can be computed with this well-known result:

$$\int_0^\pi \sin^n(x)\cos^m(x)\mathrm{d}x = \begin{cases} 0 & \text{if } m \text{ odd} \\ \frac{(n-1)!!\ (m-1)!!}{(n+m)!!}\pi & \text{if } m \text{ even and } n \text{ odd,} \\ \frac{(n-1)!!\ (m-1)!!}{(n+m)!!}2 & \text{if } n, m \text{ even.} \end{cases} \tag{B.5}$$

The second term is more cumberstone and is given by:

$$I := \int_0^\pi (\pi - x)\sin^n(x)\cos^m(x)\mathrm{d}x \tag{B.6}$$

which we solve using integration by parts with $u = \pi - x$ and $\mathrm{d}v = \sin^n(x)\cos^m(x)\mathrm{d}x$, yielding

$$I = u(0)v(0) - u(\pi)v(\pi) + \int_0^\pi v(x')\mathrm{d}x', \tag{B.7}$$

where $v(x') = \int_0^{x'} \sin^n(x)\cos^m(x)\mathrm{d}x$. This gives $v(0) = 0$ and $u(0) = 0$, simplifying $I = \int_0^\pi v(x')\mathrm{d}x'$.

We first focus on $v(x')$: for $\underline{n \text{ odd}}$, there exists a $n' \in \mathbb{N}$ so that $n = 2n' + 1$, resulting

$$v(x') = \int_0^{x'} \sin^{2n'}(x)\cos^m(x)\sin(x)\mathrm{d}x = -\int_0^{\cos(x')} (1-u^2)^{n'} u^m \mathrm{d}u \tag{B.8}$$

Where we used $\sin^2(x) + \cos^2(x) = 1$ and the substitution $u = \cos(x) \implies \mathrm{d}u = -\sin(x)\mathrm{d}x$. Using the binomial expansion, we get

$$v(x') = -\int_0^{\cos(x')} \sum_{i=0}^{n'} \binom{k}{i} (-u^2)^i u^m \mathrm{d}u = \sum_{i=0}^{n'} (-1)^{i+1} \binom{k}{i} \frac{\cos(x')^{2i+m+1} - 1}{2i+m+1}. \tag{B.9}$$

Similarly, for $\underline{m \text{ odd}}$, we have $m = 2m' + 1$ and use the substitution $u = \sin(x)$, to obtain

$$v(x') = \sum_{i=0}^{m'} (-1)^i \binom{k}{i} \frac{\sin(x')^{2i+n+1}}{2i+n+1}. \tag{B.10}$$

For $\underline{n \text{ and } m \text{ even}}$, we set $n' = n/2$ and $m' = m/2$ and use the double-angle identity, yielding

$$v(x') = \int_0^{x'} \left(\frac{1-\cos(2x)}{2}\right)^{n'} \left(\frac{1+\cos(2x)}{2}\right)^{m'} \mathrm{d}x. \tag{B.11}$$

Making use of the binomial expansion twice, we retrieve

$$v(x') = 2^{-(n'+m')} \sum_{i,j=0}^{n',m'} (-1)^i \binom{n'}{i} \binom{m'}{j} \int_0^{x'} \cos(2x)^{i+j} \mathrm{d}x. \tag{B.12}$$

Returning back to the original problem $I = \int_0^\pi v(x')\mathrm{d}x'$. Depending on the parity of n and m we need to evaluate:

$$\int_0^\pi \cos(x')^p \mathrm{d}x' = \begin{cases} \frac{(p-1)!!}{p!!}\pi & \text{if } p \text{ even} \\ 0 & \text{if } p \text{ odd,} \end{cases} \quad \text{or} \quad \int_0^\pi \sin(x')^p \mathrm{d}x' = \begin{cases} \frac{(p-1)!!}{p!!}\pi & \text{if } p \text{ even} \\ \frac{(p-1)!!}{p!!}2 & \text{if } p \text{ odd.} \end{cases} \tag{B.13}$$

For m and n even we require the solution to the double integral

$$\int_0^\pi \int_0^{x'} \cos(2x)^p \mathrm{d}x\mathrm{d}x' = \begin{cases} \frac{(p-1)!!}{p!!}\frac{\pi^2}{2} & \text{if } p \text{ even} \\ 0 & \text{if } p \text{ odd.} \end{cases} \tag{B.14}$$

Combining the above intermediate results gives the solution to eq. (B.1) for the Arc Cosine kernel. In table B.1 we list the first few eigenvalues for different dimensions and compare the analytical to the numerical computation.

Table B.1 Eigenvalues for the first-order Arc Cosine kernel eq. (3.25) computed analytically and numerically for different degrees n and dimensions d. In the experiments we set values smaller than 10^{-9} to zero.

	$d=3$		$d=5$		$d=7$	
n	numerical	analytical	numerical	analytical	numerical	analytical
0	0.375	0.375	0.352	0.352	0.342	0.342
1	0.167	0.167	0.1	0.1	0.0714	0.0714
2	0.0234	0.0234	0.00977	0.00977	0.00534	0.00534
3	−2.44e−09	−3.53e−17	1.59e−09	4.24e−17	7.79e−10	5.3e−17
4	0.000651	0.000651	0.000153	0.000153	5.34e−05	5.34e−05
5	−2.01e−09	−7.07e−17	1.86e−10	−1.01e−16	−2.11e−10	−2.52e−17
6	9.16e−05	9.16e−05	1.37e−05	1.37e−05	3.34e−06	3.34e−06
7	−1.23e−09	2.83e−16	1.53e−10	2.36e−17	−1.44e−10	−4.5e−17
8	2.29e−5	2.29e−05	2.38e−06	2.38e−06	4.26e−07	4.26e−07
9	−1.78e−10	1.7e−15	−2.19e−10	3.7e−16	3.72e−11	1.9e−16

B.2 ReLU activation function

Thanks to the simple form of the ReLU's activation shape function $\sigma(t) = \max(0, t)$, its Fourier coefficients can also be computed analytically. The integral to be solved is given by

$$\sigma_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_0^1 t\, C_n^{(\alpha)}(t)\, (1-t^2)^{\alpha-1/2} \mathrm{d}t. \tag{B.15}$$

Using Rodrigues' formula for $C_n^{(\alpha)}(t)$ in **??** and the identities in **??**, we can conveniently cancel the factor $(1-t^2)^{\alpha-1/2}$. Yielding

$$\sigma_n = \omega_d \frac{(-1)^n}{2^n} \frac{\Gamma(\alpha+\frac{1}{2})}{\Gamma(\alpha+n+\frac{1}{2})} \int_0^1 t \frac{d^n}{dt^n}\Big[(1-t^2)^{n+\alpha-1/2}\Big] \mathrm{d}t \tag{B.16}$$

Using integration by parts for $n \geq 2$ we can solve the integral [Bach, 2017, Appendix D]

$$\int_0^1 t \frac{d^n}{dt^n}\Big[(1-t^2)^{n+\alpha-1/2}\Big] \mathrm{d}t = \binom{n+\alpha-1/2}{k} (-1)^k (2k)! \text{ for } 2k = n-2 \tag{B.17}$$

$$= \frac{\Gamma(n+\alpha+\frac{1}{2})(-1)^{n/2-1}\Gamma(n-1)}{\Gamma(\frac{n}{2})\Gamma(\frac{n}{2}+\alpha+\frac{3}{2})} \tag{B.18}$$

Thus, substituting $\alpha = \frac{d-2}{2}$, yields

$$\sigma_n = \frac{\Gamma(\frac{d}{2})(-1)^{n/2-1}}{\sqrt{\pi}\, 2^n} \frac{\Gamma(n-1)}{\Gamma(\frac{n}{2})\Gamma(\frac{n}{2}+\frac{d+1}{2})}, \text{ for } n = 2, 4, 6, \ldots, \tag{B.19}$$

and $\sigma_n = 0$ for $n = 3, 5, 7, \ldots$. Finally, for $n = 0$ and $n = 1$, we obtain

$$\sigma_0 = \frac{1}{2\sqrt{\pi}} \frac{\Gamma(\frac{d}{2})}{\Gamma(\frac{d+1}{2})}, \qquad \sigma_1 = \frac{1}{2\,(d-1)} \frac{\Gamma(\frac{d}{2})\Gamma(\frac{d+1}{2})}{\Gamma(\frac{d-1}{2})\Gamma(\frac{d}{2}+1)}. \tag{B.20}$$

In table B.2 we compare the analytic expression to numerical integration using quadrature. There is a close match for eigenvalues of significance and a larger discrepancy for very small eigenvalues. In practice we set values smaller than 10^{-9} to zero.

Table B.2 Eigenvalues for the ReLU activation eq. (3.25) computed analytically and numerically for different degrees n and dimensions d. In the experiments we set values smaller than 10^{-9} to zero.

	$d = 3$		$d = 5$		$d = 7$	
n	numerical	analytical	numerical	analytical	numerical	analytical
0	0.25	0.25	0.188	0.188	0.156	0.156
1	0.167	0.167	0.1	0.1	0.0714	0.0714
2	0.0625	0.0625	0.0313	0.0312	0.0195	0.0195
3	9.08e−10	0	5.86e−10	3.37e−17	−2.05e−10	2.69e−17
4	−0.0104	−0.0104	−0.00391	−0.00391	−0.00195	−0.00195
5	−1.54e−09	0	−2.77e−10	6.75e−17	1.27e−10	5.37e−17
6	0.00391	0.00391	0.00117	0.00117	0.000488	0.000488
7	−1.44e−09	2.83e−16	−2.38e−10	1.35e−16	−9.22e−11	0
8	−0.00195	−0.00195	−0.000488	−0.000488	−0.000174	−0.000174
9	6.6e−10	1.7e−15	1.38e−10	−8.1e−16	−1.49e−11	2.15e−15

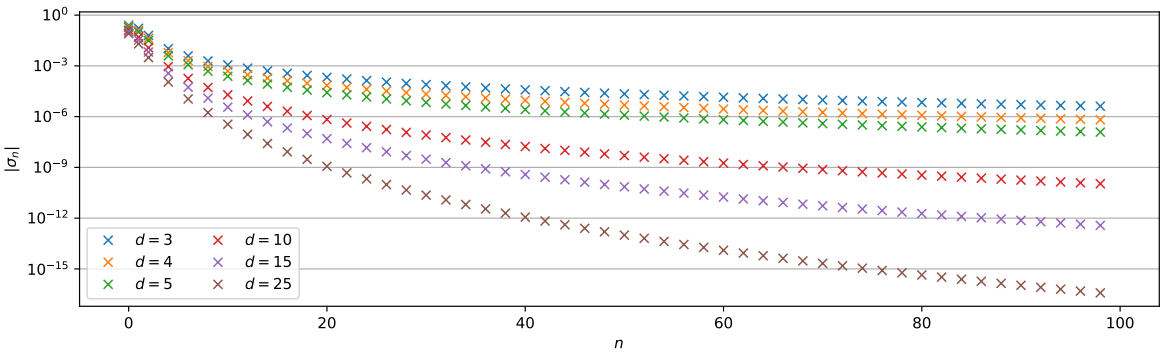


Fig. B.1 ReLU coefficients σ_n as a function of degree n for different dimensions d.