

Efficient Gaussian Processes for data-driven decision making



Vincent Dutordoir

Department of Engineering
University of Cambridge

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

Abstract

This is where you write your abstract ...

Table of contents

Chapter 1

Introduction

Using past experience to update ones behaviour is what differentiates intelligent beings from other objects in our world. To artificially create these systems is the crowning goal of the field of Artificial Intelligence (AI) and Machine Learning (ML). In Machine Learning, this is done through the manipulation of mathematical formalisms that can be translated into computer algorithms. Realising this goal will have large impact on the world and society we live in as it will free up humans from trivial tasks that require cognition, like driving cars, booking appointments, and, eventually, interpreting and acting on medical records. Though arguably the field has still a long way to go to fulfil its full potential [CITE].

A principled and elegant approach to formalise the concept of learning from current and past experience is Bayesian probability theory. In this framework one specifies beliefs over quantities of interest and updates them after observing new data. The beliefs are represented using probability distributions: the *prior* distribution describes what is known about the quantity before any data is observed, . Using Bayes' rule a prior distribution encodes . prior, likelihood and posterior. Taking actions after learning , expectations A defining attribute of Bayesian learning is the posterior distribution, and the uncertainty associated with that distribution. exploration and exploitation

The performance of a decision-making system will depend on the speed at which it can learn and the optimality of the decisions made.

A decision making system can use these posterior distributions to take actions. What makes a good decision making system uncertainty is important

In my opinion, Gaussian processes compr the right trade-offs between Studying Gaussian processes Commercial settings

In this report,

MacKay throwing the baby out with the bathwater

1.1 Contributions and Layout of this Report

This report represents my learning and the research that I conducted during the first year of my PhD degree. Most notably, we developed a novel sparse approximation for (deep) Gaussian processes based on the decomposition of the kernel in Spherical harmonics. In ?? we cover the necessary theoretical background.

Chapter 3 In this chapter we introduce a new class of inter-domain variational GPs where data is mapped onto the unit hypersphere in order to use spherical harmonic representations. The inference scheme is comparable to Variational Fourier Features, but it does not suffer from the curse of dimensionality, and leads to diagonal covariance matrices between inducing variables. This enables a speed-up in inference, because it bypasses the need to invert large covariance matrices. The experiments show that our model is able to fit a regression model for a dataset with 6 million entries two orders of magnitude faster compared to standard sparse GPs, while retaining state of the art accuracy.

The content of this chapter is largely based on:

Dutordoir2020spherical,

with the exception of the algorithm for computing the spherical harmonics in high dimensions.

Chapter 3 Following up on the previous chapter, we use the decomposition of zonal kernels to design an interdomain inducing variable that mimics the behaviour of activation functions is neural network layers.

The content of this chapter is largely based on:

dutordoir2021deep.

Chapter 4 In the last chapter of the report we will shed a light on what the future work will focus on: “Gaussian Decision Systems with Geometric Gaussian processes”.

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) [rasmussen2006] 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 [neal1996bayesian].

Following from the Kolmogorov extension theorem, we can construct a real-valued stochastic process (i.e. function) on a non-empty set \mathcal{X} , $f : \mathcal{X} \rightarrow \mathbb{R}$, 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 suffices 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 $\mu : \mathcal{X} \rightarrow \mathbb{R}$ is the mean function, which encodes the expected value of f at every x , $\mu(x) = \mathbb{E}_f[f(x)]$, and $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ 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. Moreover, $f(\mathbf{x}) \in \mathbb{R}^N$ denotes the function evaluated at a finite set of points, whereas $f^{\setminus \mathbf{x}}$ denotes another infinitely long vector similar to f but excluding $f(\mathbf{x})$. From the marginalisation property it follows that integrating out over the infinitely many points that are not included in \mathbf{x} , we obtain a valid finite-dimensional density for $f(\mathbf{x})$

$$p(f(\mathbf{x})) = \int p(f) \mathrm{d}f^{\setminus \mathbf{x}}. \quad (2.2)$$

In the case of GPs, this finite-dimensional marginal is given by a multivariate Gaussian distribution, fully characterised by the mean μ and the covariance function k

$$p(f(\mathbf{x})) = \mathcal{N}(\boldsymbol{\mu}_{\mathbf{f}}, \mathbf{K}_{\mathbf{ff}}), \quad \text{where} \quad [\boldsymbol{\mu}_{\mathbf{f}}]_i = \mu(x_i) \text{ and } [\mathbf{K}_{\mathbf{ff}}]_{i,j} = k(x_i, x_j). \quad (2.3)$$

Conditioning the GP at this finite set of points leads to a conditional distribution for $f^{\setminus \mathbf{x}}$, which is given by another Gaussian process

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

where $[\mathbf{k}_{\mathbf{f}}]_i = k(x_i, \cdot)$. The conditional distribution over the whole function $p(f | f(\mathbf{x}) = \mathbf{f})$ has the exact same form as in ???. This is mathematically slightly confusing because the random variable $f(\mathbf{x})$ is included both on the left and right-hand-side of the conditioning, but the equation is technically correct [matthews16].

2.1.1 The Beauty of Gaussian Process Regression: Exact Bayesian Inference

One of the key advantages of Gaussian processes for regression is that we can perform exact Bayesian inference. 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 $\mathbf{x} = \{x_i\}_{i=1}^N$ into a single design matrix and outputs $\mathbf{y} = \{y_i\}_{i=1}^N$ into a vector. We further assume that the data is generated by an unknown function $f : \mathcal{X} \rightarrow \mathbb{R}$, such 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.

[General introduction to Bayesian modelling] The key idea in Bayesian modelling is to specify a prior distribution over the quantity of interest. The prior encodes what we know at that point in time about the quantity. In general term, this can be a lot or a little. We encode

this information in the form of a distribution. Then, as more data becomes available, we use the rules of probability, in particular Bayes' rule, to update our prior beliefs and compute a posterior distribution (see **mackay2003information**; **bisschop** for a thorough introduction).

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, which makes 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 [**rasmussen2006**].

Following ?? the prior over function evaluations at the datapoints is defined by the covariance function k . As we assume a à-priori zero mean function μ (without loss of generality) this can be written as:

$$p(f(\mathbf{x})) = \mathcal{N}(\mathbf{0}, \mathbf{K}_{\mathbf{ff}}), \quad \text{where} \quad [\mathbf{K}_{\mathbf{ff}}]_{i,j} = k(x_i, x_j). \quad (2.5)$$

Given the function f the likelihood factorises over datapoints and is given by a Gaussian:

$$p(\mathbf{y} | f) = \prod_{i=1}^N p(y_i | f) = \prod_{i=1}^N \mathcal{N}(y_i | f(x_i), \sigma^2) \quad (2.6)$$

We can obtain the posterior over the function using Bayes' rule and the marginalisation property

$$p(f | \mathbf{y}) = \frac{p(f) p(\mathbf{y} | f)}{p(\mathbf{y})} \quad (2.7)$$

$$= p(f^{\setminus x} | f(x)) \frac{p(f(x)) \prod_{i=1}^N \mathcal{N}(y_i | f(x_i), \sigma^2)}{p(\mathbf{y})} \quad (2.8)$$

$$= \mathcal{GP}(\mathbf{k}_{\mathbf{f}}^{\top} \mathbf{K}_{\mathbf{ff}}^{-1} \mathbf{f}, \quad k(\cdot, \cdot) - \mathbf{k}_{\mathbf{f}}^{\top} \mathbf{K}_{\mathbf{ff}}^{-1} \mathbf{k}_{\mathbf{f}}), \quad (2.9)$$

The marginal likelihood (model evidence)

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y} | \mathbf{0}, \mathbf{K}_{\mathbf{ff}} + \sigma^2 \mathbf{I}) \quad (2.10)$$

1. Plot: Prior, Data, Posterior
2. occam's razor

Problems A common criticism for GPs is that any modification to this approach breaks the Gaussian assumption.

1. Non-Gaussian likelihoods
2. Large datasets
3. Transformations: log or square transform

Solutions

1. Laplace
2. Expectation Propagation
3. Sparse Variational Inference

2.2 Approximate Inference with Sparse Gaussian Processes

1. General introduction to Variational inference [blei2017variational] variational inference (VI), where the problem of Bayesian inference is cast as an optimization problem—namely, to maximize a lower bound of the logarithm of the marginal likelihood.
2. Sparse approximations [Snelson05; quinonero2005unifying]

2.3 Interdomain Inducing Variables

The basis functions used in the approximate posterior mean TODO ref are determined by the covariance between the inducing variables and other function evaluations $[c_u(\cdot)]_m = \text{Cov}(f(\cdot), u_m)$. Commonly, the inducing variables are taken to be function values $u_m = f(w_m)$, which leads to the kernel becoming the basis function $[c_u(\cdot)]_m = k(\mathbf{w}_m, \cdot)$. *Interdomain* inducing variables [lazaro2009inter] select different properties of the GP (e.g. integral transforms $u_m = \int f(\mathbf{x})g_m(\mathbf{x})d\mathbf{x}$), which modifies this covariance (see [van2020framework; Leibfried2020Tutorial] for an overview), and therefore gives control over the basis functions. Most current interdomain methods are designed to improve computational properties [hensman2017variational; burt2020variational; Dutordoir2020spherical]. Our aim is to control $c_u(\cdot)$ to be a typical NN activation function like a ReLU or Softplus. This was also investigated by sun2021neural, although they found less common saturating activation functions.

2.3.1 Example: heavyside inducing variable

$f \sim \mathcal{GP}$ defined on \mathbb{S}^1 (the unit circle), $f : [-\pi, \pi] \rightarrow \mathbb{R}, \theta \mapsto f(\theta)$.
kernel (Arc Cosine order 0):

$$k(\theta, \theta') = \kappa(\rho) = \pi - |\theta - \theta'| \quad (2.11)$$

$$\frac{d}{d\theta} \Big|_{\theta=\theta_m} k(\theta, \theta') = \quad (2.12)$$

$$u_m = \mathcal{L}_m(f) \quad (2.13)$$

$$\mathcal{L}_m = \frac{d}{d\theta} \Big|_{\theta=\theta_m} + \int d\theta \quad (2.14)$$

$$\text{Cov}(u_m, f(\theta')) = \mathbb{E}_f[\mathcal{L}_m(f) f(\theta')] \quad (2.15)$$

$$= \mathcal{L}_m k(\theta', \cdot) \quad (2.16)$$

$$= \frac{d}{d\theta} \Big|_{\theta=\theta_m} k(\theta', \theta) + \int_{-\pi}^{\pi} k(\theta', \theta) d\theta \quad (2.17)$$

2.4 Deep Gaussian Processes

dutoir2021gpflux

2.5 Kernels and Reproducing Kernel Hilbert Space

Kernels have a rich history in machine learning and functional analysis. Kernels became widely noticed by the ML community when linear SVMs [CITE] were kernelised. This means that rather than using x^\top , one uses a function $k(x, x')$, which corresponds to an inner-product in some space $\phi(x)^\top \phi(x')$. This concept is also referred to as the “kernel trick” in the literature. This made SVMs a compelling model for non-linear modelling problems. Prior to the deep learning revolution that began in the late nineties, kernel SVMs set the state of the art results in many tasks: XXX. Kernels are also used in splines, Principle Component Analysis (PCA), Gaussian processes and in general allow for the development of versatile algorithms to analyse and process data. Kernels operate through pairwise comparisons on datapoints and are versatile because they do not impose strong assumptions regarding the type of data, such as vectors, graphs or strings, they operate on. In what follows, we are going to focus on the theoretical properties of Mercer kernels with the application of Gaussian processes in mind, and the infinite vector space these kernels characterise. This background material will be important for the remaining chapters of the thesis. The inspiration for this section came from course Julian Marial and review paper on XXX.

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

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

One of the simplest examples of a p.d. kernel is $k(x, x') = xx'$, where $\mathcal{X} = \mathbb{R}$. This kernel is often called the linear kernel. While being simple, as we will see through the next few theorems, the associated space of functions induced by this kernel is often too restrictive. In

the next theorems we will see how kernels characterise a space over functions. Some of these spaces will be very explicit. For example, we will show how the linear kernel leads to a space of functions that are also linear, i.e. of the form $x \mapsto ax + b$. Other spaces belonging to kernel will be defined in a more abstract way.

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

$$k(x, x') = \langle \phi(x), \phi(x') \rangle_{\mathcal{H}} \quad (2.18)$$

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 arbitrarily close as the sequence progresses. Intuitively, 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 ???. It is a space over function from \mathcal{X} to \mathbb{R} that 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 it more explicit we also write $\phi(x) = k_x$ to highlight that k_x is another function in the space $\mathcal{X} \rightarrow \mathbb{R}$.

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 $k : \mathcal{X} \times \mathcal{X} \rightarrow \mathbb{R}$ is called a reproducing kernel of \mathcal{H} if*

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

2.

1. History: RKHS pure math and into machine learning for SVM, splines and

2. Positive Definite and Symmetry

3. RKHS

4. Bochner’s theorem

5. Mercer Decomposition

6. Examples of RKHS

7. RKHS through Spectral Decomposition

8. Representer Theorem

9. Show how sparse approximation links anchor points

2.6 Spectral Representation of Gaussian Processes

Chapter 3

Spherical Harmonic Variational Gaussian Processes

3.1 ArcCosine kernel and Associated Reproducing Hilber Kernel Space

The first order Arc Cosine kernel mimics the computation of infinitely wide fully connected layers with ReLU activations. **cho2009kernel** showed that for $\sigma(t) = \max(0, t)$, the covariance between function values of $f(\mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x})$ for $\mathbf{w} \sim \mathcal{N}(0, d^{-1/2} \mathbf{I}_d)$ and $\mathbf{w} \in \mathbb{R}^d$ is given by

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}_{\mathbf{w}} [\sigma(\mathbf{w}^\top \mathbf{x}) \sigma(\mathbf{w}^\top \mathbf{x}')] = \underbrace{\|\mathbf{x}\| \|\mathbf{x}'\|}_{\text{radial}} \underbrace{\frac{1}{\pi} (\sqrt{1-t^2} + t(\pi - \arccos t))}_{\text{angular (shape function) } s(t)}, \quad (3.1)$$

where $t = \frac{\mathbf{x}^\top \mathbf{x}'}{\|\mathbf{x}\| \|\mathbf{x}'\|}$. The factorisation of the kernel in a radial and angular factor leads to an RKHS consisting of functions of the form $f(\mathbf{x}) = \|\mathbf{x}\| g(\frac{\mathbf{x}}{\|\mathbf{x}\|})$, where $g(\cdot)$ is defined on the unit hypersphere $\mathbb{S}^{d-1} = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\|_2 = 1\}$ but fully determines the function on \mathbb{R}^d .

The shape function can be interpreted as a kernel itself, since it is the restriction of $k(\cdot, \cdot)$ to the unit hypersphere. Furthermore its expression only depends on the dot-product between the inputs so it is a zonal kernel (also known as a dot-product kernel [**bietti2020deep**]). This means that the eigenfunctions of the angular part of $k(\cdot, \cdot)$ are the spherical harmonics $\phi_{n,j}$ (we index them with a level n and an index within each level $j \in \{1, \dots, N_n^d\}$) [**wendland2005; Dutordoir2020spherical**]. Their associated eigenvalues only depend on n :

$$\lambda_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{-1}^1 s(t) C_n^{(\alpha)}(t) (1-t^2)^{\frac{d-3}{2}} dt, \quad (3.2)$$

where $C_n^{(\alpha)}(\cdot)$ is the Gegenbauer polynomial¹ of degree n , $\alpha = \frac{d-2}{2}$, ω_d is a constant that depends on the surface area of the hypersphere. Analytical expressions of λ_n are provided in ???. The above implies that k admits the Mercer representation:

$$k(\mathbf{x}, \mathbf{x}') = \|\mathbf{x}\| \|\mathbf{x}'\| \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} \lambda_n \phi_{n,j} \left(\frac{\mathbf{x}}{\|\mathbf{x}\|} \right) \phi_{n,j} \left(\frac{\mathbf{x}'}{\|\mathbf{x}'\|} \right), \quad (3.3)$$

and that the inner product between any two functions $g, h \in \mathcal{H}$ is given by:

$$\langle g, h \rangle_{\mathcal{H}} = \sum_{n,j} \frac{g_{n,j} h_{n,j}}{\lambda_n} \quad (3.4)$$

where $g_{n,j}$ and $h_{n,j}$ are the Fourier coefficients of g and h , i.e. $g(\mathbf{x}) = \sum_{n,j} g_{n,j} \|\mathbf{x}\| \phi_{n,j}(\mathbf{x})$.

3.2 Spherical Harmonics

Spherical harmonics are a special set of functions defined on the hypersphere and play a central role in harmonic analysis and approximation theory [wendland2005]. 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 [frye2014; dai2013], 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 orthonormal 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 accompanied 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 dai2013; frye2014 for a comprehensive overview.

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

$$\langle f, g \rangle_{L_2(\mathbb{S}^{d-1})} = \frac{1}{\Omega_{d-1}} \int_{\mathbb{S}^{d-1}} f(x) g(x) d\omega, \quad (3.5)$$

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

$$\Omega_{d-1} = \int_{\mathbb{S}^{d-1}} d\omega(x) = \frac{2\pi^{d/2}}{\Gamma(d/2)}. \quad (3.6)$$

¹See ?? for a primer on Gegenbauer polynomials and spherical harmonics.

Throughout this section we use the following notation and definitions. For $x = (x_1, \dots, x_d) \in \mathbb{R}^d$ and $\alpha = (\alpha_1, \dots, \alpha_d) \in \mathbb{N}^d$, a monomial x^α is a product $x^\alpha = x_1^{\alpha_1} \dots x_d^{\alpha_d}$, which has degree $|\alpha| = \alpha_1 + \dots + \alpha_d$. 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 $f : \mathbb{R}^d \rightarrow \mathbb{R}$ is said to be *harmonic* if $\Delta f = 0$, where $\Delta = \partial_{x_1}^2 + \dots + \partial_{x_d}^2$ and ∂_{x_i} the partial derivate w.r.t. the i -th variable.

Definition 3. *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*

$$\mathcal{H}_n^d = \{p \in \mathcal{P}_n^d : \Delta p = 0 \text{ and } p : \mathbb{S}^{d-1} \rightarrow \mathbb{R}\}. \quad (3.7)$$

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

$$\dim(\mathcal{H}_n^d) = \frac{2n+d-2}{n} \binom{n+d-3}{d-1} := N_n^d. \quad (3.8)$$

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

$$\mathcal{H}_n^d = \text{span}(\phi_{n,1}, \dots, \phi_{n,N_n^d}), \quad \text{and} \quad \langle \phi_{n,j}, \phi_{n',j'} \rangle_{L_2(\mathbb{S}^{d-1})} = \delta_{nn'} \delta_{jj'}. \quad (3.9)$$

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 [frye2014]. This means that we can decompose a function $f : \mathbb{S}^{d-1} \rightarrow \mathbb{R}$ as

$$f = \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} \hat{f}_{n,j} \phi_{n,j}, \quad \text{with} \quad \hat{f}_{n,j} = \langle f, \phi_{n,j} \rangle_{L_2(\mathbb{S}^{d-1})}, \quad (3.10)$$

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 $n = 0, 1, 2, \dots$ denotes the degree (or level) and $j = 1, \dots, N_n^d$ 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 differential 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 **dai2013**, which consists of a product over polynomials. This makes the implementation cumbersome and numerically unstable, and only practically useful up to 10 dimensions [**Dutordoir2020spherical**]. 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 2 (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 $C_n^{(\alpha)} : [-1, 1] \rightarrow \mathbb{R}$ of degree n can be written as*

$$\sum_{j=1}^{N_n^d} \phi_{n,j}(x) \phi_{n,j}(x') = \frac{n + \alpha}{\alpha} C_n^{(\alpha)}(x^\top x') \quad \text{with} \quad \alpha = \frac{d-2}{2}. \quad (3.11)$$

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 geodesic distance between η and x .

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

$$\frac{1}{\Omega_{d-1}} \int_{\mathbb{S}^{d-1}} f(\eta^\top x) \phi_{n,j}(x) d\omega(x) = \lambda_n \phi_{n,j}(\eta), \quad (3.12)$$

where λ_n is a constant defined by

$$\lambda_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{-1}^1 f(t) C_n^{(\alpha)}(t) (1-t^2)^{\frac{d-3}{2}} dt, \quad (3.13)$$

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

3.2.1 Zonal Spherical Harmonics

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

Theorem 4. *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*

$$\det \begin{bmatrix} C_n^{(\alpha)}(1) & \dots & C_n^{(\alpha)}(\eta_1^\top \eta_M) \\ \vdots & & \vdots \\ C_n^{(\alpha)}(\eta_M^\top \eta_1) & \dots & C_n^{(\alpha)}(1) \end{bmatrix} > 0. \quad (3.14)$$

Finding a basis for \mathcal{H}_n^d is thus equivalent to finding a set of N_n^d points that satisfy ???. Crucially, **dai2013** show that there always exists a fundamental system of degree n and N_n^d points.

Following the theorem, 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 ???. 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

$$\mathbf{M}(\boldsymbol{\eta}, \eta_*) = \left[\begin{array}{c|c} C_n^{(\alpha)}(\boldsymbol{\eta}\boldsymbol{\eta}^\top) \in \mathbb{R}^{M \times M} & C_n^{(\alpha)}(\boldsymbol{\eta}\eta_*^\top) \in \mathbb{R}^{M \times 1} \\ \hline C_n^{(\alpha)}(\boldsymbol{\eta}\eta_*^\top)^\top \in \mathbb{R}^{1 \times M} & C_n^{(\alpha)}(1) \in \mathbb{R} \end{array} \right], \quad (3.15)$$

where $C_n^{(\alpha)}(\boldsymbol{\eta}\boldsymbol{\eta}^\top)$ corresponds to elementwise evaluating the Gegenbauer polynomial $C_n^{(\alpha)} : [-1, 1] \rightarrow \mathbb{R}$ 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

$$\eta = \operatorname{argmax}_{\eta_* \in \mathbb{S}^{d-1}} \det(\mathbf{M}(\boldsymbol{\eta}, \eta_*)), \quad (3.16)$$

in order to satisfy the condition in ??. 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

$$\eta = \operatorname{argmin}_{\eta_* \in \mathbb{R}^d} C_n^{(\alpha)}\left(\frac{\eta_*}{\|\eta_*\|} \boldsymbol{\eta}^\top\right) \left[C_n^{(\alpha)}(\boldsymbol{\eta}\boldsymbol{\eta}^\top) \right]^{-1} C_n^{(\alpha)}\left(\boldsymbol{\eta} \frac{\eta_*^\top}{\|\eta_*\|}\right). \quad (3.17)$$

The complete algorithm is given in ??.

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**
 $\eta = \operatorname{argmax}_{\eta_* \in \mathbb{R}^d} \det(\mathbf{M}(\boldsymbol{\eta}, \frac{\eta_*}{\|\eta_*\|}))$ // Using a local optimisation method (e.g.,
 BFGS) and ??
 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 5. 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

$$\phi_{n,j}(x) = \sum_{i=1}^{N_n^d} \mathbf{L}_{j,i} C_n^{(\alpha)}(\eta_i^\top x) \quad (3.18)$$

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

Chapter 4

Deep Neural Networks as Point Estimates for Deep Gaussian Processes

mackay1992practical; **mackay1992bmc** noted very early that the Bayesian treatment of neural networks had large potential. To start, it can help to quantify estimates of the error bars on the network outputs, but it can also provide an objective that can be used for the comparison of alternative network architectures. The reality, however, was that Bayesian inference in neural networks was –and still is– challenging, and in practice one has to resort to either crude approximations (e.g., Laplace approximation [**MacKay1998laplace**]) or lengthy computations (e.g., Markov Chain Monte Carlo [**neal1992bayesian**]).

Building on the foundational work of MacKay, and driven by the amazing successes of large deterministic deep neural networks (DNNs), the literature has seen an upspring in the development of more scalable methods to perform approximate Bayesian inference in DNNs [**blundell2015**; **Kingma2015local**; **Gal2016dropout**]. However, it remains challenging to encode prior assumptions on functions through distributions on weights and the large number of parameters to be estimated makes it computationally prohibitive. Furthermore, the strong approximations used both during modelling and inference make it unclear to what extent these models approximate the true posterior distribution [**hron2017variational**; **foong2019expressiveness**]. They also do not deliver on an important promise of Bayesian methods: an approximate marginal likelihood objective that can be used for automatic model selection and hyperparameter learning. A different approach may thus be necessary to unlock the Bayesian benefits in deep learning.

neal1996bayesian showed that for infinitely wide single-layer BNNs the distribution over the non-linear functions are given by Gaussian processes. **williams1998computation**; **cho2009kernel** extended this theory and derived the kernel corresponding to an infinite-width

BNN with Sigmoidal and ReLU activation function, respectively. The beauty of this connection is that performing Bayesian inference in the corresponding GP model can be done exactly and analytically – all in a single elegant framework [rasmussen2006]. Since, various approximations to the exact GP framework have been developed to allow for non-Gaussian likelihoods [kuss2005assessing; hensman2013], large datasets [hensman2015scalable; wang2019exact], and even neural network like structures such as convolutions [vanderwilk2017conv]. Crucially, the approximations to the marginal likelihood still enable the main Bayesian benefits: model uncertainty and learning model hyperparameters (e.g., [vanderwilk2018invariances; Dutordoir2020convolutional]).

More recently, matthews2018gaussian discovered the equivalence between *deep* (i.e. multi-layer) BNNs and GPs. This has led to the development of deep (fully connected and convolutional) neural network kernels for GPs (NN-GPs). Interestingly, the performance of the non-Bayesian neural nets significantly outperforms the corresponding GPs [garriga2018deep; novak2018bayesian]. The discrepancy hints at the fact that these single-layer GPs, even when configured with very expressive DNN equivalent kernels, are missing a crucial component: the ability to learn feature representations from the data.

Deep Gaussian processes [Damianou2013] are an interesting avenue to tackle these challenges. They are built by hierarchically stacking GPs on top of each other, which enables the learning of more powerful representations through compositional features. Moreover, their Bayesian approximations in function-space seem to be of higher quality than those of weight-space BNNs, e.g. as supported by the successful use of marginal likelihood estimates for hyperparameter learning [Damianou2013].

While promising, DGPs have struggled for relevance in applications due to the challenges and cost associated with Bayesian inference. Training DGPs is computationally expensive and requires very careful setting of the parameters. Furthermore, the hierarchical prior induced by naively stacking stationary kernels gives rise to pathological, “collapsed” samples duvenaud2014avoiding. Considerable progress for scalable inference in DGPs was made by hensman2014nested; salimbeni2017doubly, who derived stochastic variational lower bounds. The formulation of these bounds closely resembles the computations of training a feed-forward neural network with regularisation terms, and has greatly inspired this work.

Building on salimbeni2017doubly and to further improve the scalability of DGPs, cutajar2017random used a Random Fourier Feature [rahimi2008random] approximation of the kernel. While successful, this approach introduces an approximation in both the prior and the posterior of the model. More recently, rudner2020inter proposed Fourier features of the Matérn RKHS, following hensman2017variational, to build inter-domain DGPs. Like for single layer VFF models, this approach can lead to faster training and improved performance, but is only computationally feasible for data of dimension one or two. In parallel with our work, sun2021neural explored the idea of using single-layer neural networks to parameterise inducing points in shallow GPs. Similar to this paper, their method makes use of the spherical

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

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

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

4.1 Method: Gaussian Process Layers with Activated Inducing Variables

The concepts introduced in the background section can now be used to summarise our approach: We consider DGP models with GP layers that have an Arc Cosine kernel and Variational Fourier Feature style inducing variables $u_m = \langle f(\cdot), g_m(\cdot) \rangle_{\mathcal{H}}$ [hensman2017variational]. We then choose inducing functions $g_m(\cdot)$ that have the same shape as neural network activation functions (??). This yields basis functions for the SVGP model that correspond to activation functions (??), and thus to a model whose mean function can be interpreted as a classic feed forward neural network. ?? covers the mathematical intricacies associated with the construction described above.

4.1.1 Activated Inducing Functions and their Spherical Harmonic Decomposition

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

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

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

$$g_m : \mathbb{R}^d \rightarrow \mathbb{R}, \quad \mathbf{x} \mapsto \|\mathbf{x}\| \|\mathbf{w}_m\| \sigma\left(\frac{\mathbf{w}_m^\top \mathbf{x}}{\|\mathbf{w}_m\| \|\mathbf{x}\|}\right), \quad (4.1)$$

with $\mathbf{w}_m \in \mathbb{R}^d$ a parameter, and $\sigma : [-1, 1] \rightarrow \mathbb{R}$ the function that determines the value of $g_m(\cdot)$ on the unit hypersphere. In ??, we show that choosing $\sigma(\cdot)$ to be a ReLU ($\sigma(t) = \max(0, t)$) or a Softplus ($\sigma(t) = \log(1 + \exp(3t))$) 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 $\mathbf{w}_m \in \mathbb{R}^d$ 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 ?? is particularly convenient when it comes to representing $g_m(\cdot)$ in the basis of the eigenfunctions of \mathcal{H} , which is required for computing inner products. It indeed allows us to make use of the Funk-Hecke theorem (see ??) and to eventually obtain

$$g_m(\mathbf{x}) = \|\mathbf{x}\| \|\mathbf{w}_m\| \sum_{n=0}^{\infty} \sum_{j=1}^{N_n^d} \sigma_n \phi_{n,j}\left(\frac{\mathbf{w}_m}{\|\mathbf{w}_m\|}\right) \phi_{n,j}\left(\frac{\mathbf{x}}{\|\mathbf{x}\|}\right), \quad (4.2)$$

$$\text{where} \quad \sigma_n = \frac{\omega_d}{C_n^{(\alpha)}(1)} \int_{-1}^1 \sigma(t) C_n^{(\alpha)}(t) (1 - t^2)^{\frac{d-3}{2}} dt.$$

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

4.1.2 Activated Interdomain Inducing Variables

We define our *activated* interdomain inducing variables as

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

Since the GP samples do not belong to the RKHS there are mathematical subtleties associated with such a definition, which are detailed in ??. Assuming for now that they are indeed well defined, using these interdomain inducing variables as part of the SVGP framework requires access to two quantities: (i) their pairwise covariance, and (ii) the covariance between the GP

and the inducing variables. The pairwise covariance, which is needed to populate \mathbf{C} , is given by

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

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

$$\text{Cov}(u_m, f(\mathbf{x})) = \langle k(\mathbf{x}, \cdot), g_m(\cdot) \rangle_{\mathcal{H}} = g_m(\mathbf{x}) \quad (4.5)$$

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

4.1.3 Analysis of the interplay between kernels and inducing functions

In this section we describe the mathematical pitfalls that can be encountered [sun2021neural] when defining new inducing variables of the form of ??, and how we address them. We discuss two problems: 1) the GP and the inducing function are not part of the RKHS, 2) the inducing functions are not expressive enough to explain the prior. Both problems manifest themselves in an approximation that is overly smooth and over-estimates the predictive variance.

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

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

Using this expression to compute the RKHS norm of a GP sample $f(\cdot)$ yields $\|f\|^2 = \sum_{n,j} \xi_{n,j}^2$, which is a diverging series. This is a clear indication that the GP samples do not belong to the RKHS [kanagawa2018gaussian], and that expressions such as $\langle f(\cdot), g(\cdot) \rangle_{\mathcal{H}}$ should be manipulated with care. According to the definition given in ??, the RKHS inner product is a series that converges if computed for any two elements $g(\cdot), h(\cdot) \in \mathcal{H}$. The inner product operator can however be extended to the case where one input lives in a space larger than \mathcal{H} , provided that restrictions are introduced on the second input to guarantee the convergence of the series. In other words, even if the decay of the Fourier coefficients of $f(\cdot)$ is too slow to

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

The above reasoning indicates that, for a given kernel $k(\cdot, \cdot)$, some activation functions $g_m(\cdot)$ will result in inducing variables $u_m = \langle f(\cdot), g_m(\cdot) \rangle_{\mathcal{H}}$ that are well defined whereas other activation functions do not. For example, if we consider the case of the Arc Cosine kernel and the ReLU inducing function, the decay rate of σ_n is proportional to the square root of λ_n [bach2017breaking; bietti2020deep]. This implies that the inner product series diverges and that this kernel and inducing variable cannot be used together. Alternatively, using smoother activation functions for $\sigma(t)$, such as the Softplus, results in a faster decay of the coefficients σ_n and can guarantee that inducing variables are well defined.

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

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

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

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

4.2 Experiments

We have shown that that we can build SVGP models with basis functions that behave like neural net activations. This equivalence has the practical advantage that we can train the means of the GP layers in our DGP as if they are a neural network model — making use of the great advances made by the deep learning community. Once the mean of the DGP is trained, we can further optimise the remaining model hyper- and variational parameters w.r.t. the ELBO, which is a more principled objective [Fong2019On]. This approach allows us to exploit the benefit of both, the efficient training of the DNN in combination with the principled uncertainty estimate of the DGP. For all SVGP models we use the Arc Cosine kernel and inducing variables obtained with the Softplus activation (??) with $\tilde{N} = 20$, for the reasons explained in ??.

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 [Gal2016dropout] neural networks in terms of calibrated uncertainty. We acknowledge that the NNs we benchmark against are the models for which we can build an equivalent DGP. While this leads to a fair comparison, it excludes recent improvements such as Transformer and Batch Norm layers.

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

Regression on UCI benchmarks We compare a suit of models on a range of regression problems. The important aspect is that we keep the model configuration and training procedure fixed across datasets. We use three-layered models with 128 inducing variables or hidden units. In each layer, the number of output heads is equal to the input dimensionality of the data. The Activated DGP (ADGP) and neural network approaches (Vanilla and Dropout) use Softplus activation functions. The Dropout model [Gal2016dropout] 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.

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

Large scale image classification In this experiment we measure the performance of our models under dataset shifts [Ovadia2019]. 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 [he2016deep] to extract useful features before passing them to our dense GP or NN layers. Details of the model architectures are given in ?. As previously, the ADGP model is initialised to the solution of the Vanilla model, and training is then continued using the ELBO. In ?? we observe that the models perform very similar in terms of prediction accuracy, but that ADGP better account for uncertainty as evidenced by the Test Log Likelihood metric.

Chapter 5

Future Research

5.1 Gaussian Decision Systems in Non-Euclidian Spaces

To date, my research has focussed on pushing Gaussian processes (GPs) into the realm of deep learning. I have developed Deep Convolutional Gaussian processes that mimic the convolutional and layared architectures, Conditional density estimation model which are similar to Variational Auto Encoders, and more recently I have worked on Deep Gaussian processes that have activation functions as basis functions. This research has been fruitful and I believe we have advanced the field of approximate Bayesian inference in deep compositional models, and showed how deep Gaussian processes are an interesting non-parametric alternative to Bayesian neural networks.

In what comes next, I want to be more clear about the differences of neural networks and Gaussian processes, and the regimes in which they thrive. For example, NNs can handle—and in practice require—very large datasets, whereas Gaussian process models are more comfortable in the small data regime. Neural networks, in the presence of large datasets, are extremely good at learning integrate latent representations in very non-linear tasks, as evidenced by the latest models in language modelling and image . Non-parametric Gaussian processes, however, non-parametric nature makes exact Bayesian inference also too expensive. Defining a kernel is extremely difficult and Bayesian inference too costly key in uncertainty.

Uncertainty is important in many situations, as it hyperparmaeter learning, but it is most vital in decision making systems. A model that can represent what is does not know is as valuable as being accurate most of the time. This is especially the case when your model is part of system that drives decisions that put expensive resources at stake and only few historical datapoints are available. Example: kidney transplation algo.

Problem setting

1. Black Box Functions $f : \mathcal{X} \rightarrow \mathcal{Y}$

2. We want to estimate a computable property $\mathcal{O}_{\mathcal{A}}(f)$
3. \mathcal{A} is an algorithm $\mathcal{O}_{\mathcal{A}}(f) = \mathcal{A}(f)$
4. Evaluating f is *very* expensive (we can only evaluate it a limited amount of times)

Examples

1. Bayesian Optimisation: $\mathcal{A}(f) = \operatorname{argmax}_{x \in \mathcal{X}} f(x)$, which implies $\mathcal{O}_{\mathcal{A}}(f) = x^*$.
2. Sensor Placement (Active Learning): $\mathcal{O}_{\mathcal{A}}(f) = \operatorname{argmax}_{X \subset \mathcal{X}, |X|=T} \operatorname{MI}(f, f(X))$.
3. Level sets: $\mathcal{O}_{\mathcal{A}}(f) = \{X \subset \mathcal{X} : f(x) > C, \forall x \in X\}$.
4. Shortest path: $\mathcal{O}_{\mathcal{A}}(f) = \text{shortest path between two nodes in a graph}$.

Model We model f by a Gaussian process

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

1. Low dimensions
2. Prior knowledge
3. Limited and very expensive data

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

Objectives

1. Theory and analysis Theory – test of time by Krausse paper UCB regret bounds Although dependant on the prior Kidney matching algorithm
2. Getting these ideas into people’s hands to get applications off the ground
 - (a) Graph GPs for combinatorial optimization use cases
 - (b) Manifold GPs for scientific use cases

Collaborators

1. Alex Terenin (Imperial College London)
2. Willie Neiswanger (Stanford University)

5.2 Projects in Progress

1. “Pay Attention to Deep Gaussian Processes”

Transformer Layer Gaussian Processes using an explicit feature representation of the attention operation.

$$\exp(\mathbf{x}^\top \mathbf{y}) = \Phi^\top(\mathbf{x})\Phi(\mathbf{y})$$

2. A Unifying Theory for Interdomain Gaussian Processes.
3. VISH-PI: Probabilistic Integration with Variational Inducing Spherical Harmonics.

