Spectral Methods in Gaussian Modelling

Contents

I Introduction

I.1 Probabilistic Modelling

Modelling is central to science: it provides insights into phenomena, helps us make predictions, and allows us to test hypotheses. An important aspect of modelling is the ability to quantify uncertainty, e.g. uncertainty in forecasts or uncertainty about what the model parameters should be. The probabilistic approach to modelling expresses uncertainty with the language of probability theory. Bayesian modelling, in this context, is synonymous with the probabilistic approach in that it uses probability theory estimate model parameters from data, make forecasts, and test hypotheses.

Let us clear up some common misconceptions about Bayesian modelling. Fundamentally, Bayesian modelling uses probability theory—the sum rule and product rule in particular—to represent and manipulate all forms of uncertainty in the model. These two rules yield Bayes' rule. Bayes' rule tells us how to update our uncertainty about *any* aspect of our model, say the random variable θ, in light of newly observed data X:

$$P(\theta \mid X) = \frac{P(X \mid \theta) P(\theta)}{P(X)}.$$

Here the *prior* $P(\theta)$, our belief about θ before observing any data, is updated the *posterior* $P(\theta \mid X)$, our belief about θ after observing X. This requires us to explicitly specify a prior $P(\theta)$ before we observe any data, which is one of the common objections to Bayesian modelling. Specifying the prior $P(\theta)$ should be viewed as stating our assumptions about the model. Crucially, all models make assumptions, and, instead of hiding these assumptions, the Bayesian approach uses the language of probability to make them explicit. Another common objection is that model parameters are treated as random variables—that a true set of parameters exist, so it does not make sense to treat them as random variables. This too is a misunderstanding. In the Bayesian framework, probabilities are used to represent *our belief* of the parameters θ. Whether or not some true, fixed set of parameters θ exist is a different matter; $P(\theta)$ just represents our understanding of what θ might be.

In this framework, modelling becomes a simple two step procedure: (1) write down your assumptions with the language of probability theory and (2) once data is observed, use the rules of probability theory to make inferences about unknown quantities in

the model. This procedure decouples modelling and making predictions, which often become entangled when the goal is to make forecasts: first build a model that encodes your assumptions and then use this model to make predictions.

Stemming from their interpretability, modularity, and tractability, Gaussian processes (GPs) form a powerful and popular framework for probabilistic modelling (Rasmussen et al., 2006). Gaussian processes are successfully applied in a wide variety of contexts: they can be used to automatically discover structure in signals (David Duvenaud, 2014), are state of the art in numerous regression tasks (T. D. Bui et al., 2016b), provide data-efficient models in reinforcement learning (Deisenroth et al., 2011), and find many applications in probabilistic numerics (Hennig et al., 2015), such as in optimisation (Brochu et al., 2010) and quadrature (Minka, 2000).

Contrary to parametric models, there is no finite number of parameters that parametrise a Gaussian process. Instead, the model complexity of a Gaussian process grows as evidence is accumulated; such models are called *nonparametric* models. This property allows Gaussian processes to learn complex functions if plenty of evidence is available, and conversely makes them robust against overfitting if only little evidence is at hand.

Gaussian processes achieve this automatic calibration of model complexity by placing a prior distribution directly on the space of functions. They can be thought of as the infinite-dimensional generalisation of a multivariate normal distribution. In same the way as a multivariate normal is parametrised by a mean vector and covariance matrix, a Gaussian process is parametrised by a mean *function* and a covariance *function*—the latter is also often called the *kernel*. The kernel of a Gaussian process determines the properties of the functions that are supported by the induced prior on the space of functions. Consequently, it is crucial to choose the kernel correctly. But choosing an appropriate kernel is often difficult, where even experts use a little bit of black magic.

This reveals the first issue with Gaussian processes, which is the choice of kernel; we refer to this problem as the *kernel design problem*. The second issue with Gausssian processes is that they are computationally expensive. For N observations, inference and learning take $O(N^3)$ time and $O(N^2)$ memory. These complexities are even worse in the multi-output setting: if every observation has P outputs, then inference and learning take $O(N^3P^3)$ time and and $O(N^2P^2)$ memory.

I.2 Gaussian Processes

A Gaussian process is an infinite collection of real-valued random variables $(f(t))_{t\in\mathcal{T}}$ indexed on some set $\mathcal{T}$ such that all finite-dimensional distributions[1] (f.d.d.'s) are Gaussian. Often, $\mathcal{T} = \mathbb{R}$ is time, or $\mathcal{T} = \mathbb{R}^D$ is some feature space. These f.d.d.'s must be consistent in the sense that they respect marginalisation and permutations of the variables. The f.d.d.'s then uniquely extend to a distribution on the space of functions, a consequence of the Kolmogorov Extension Theorem.

The mean function $m : \mathcal{T} \to \mathbb{R}$ and kernel $k : \mathcal{T} \times \mathcal{T} \to \mathbb{R}$ of a Gaussian process specify the mean and covariance of the f.d.d.'s, in the following way: if $(t_1, \ldots, t_N) \in \mathcal{T}^n$, then

$$\begin{bmatrix} f(t_1) \\ \vdots \\ f(t_N) \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m(t_1) \\ \vdots \\ m(t_N) \end{bmatrix}, \begin{bmatrix} k(t_1, t_1) & \cdots & k(t_1, t_N) \\ \vdots & \ddots & \vdots \\ k(t_N, t_1) & \cdots & k(t_N, t_N) \end{bmatrix} \right).$$

The kernel k must be such every such covariance matrix is a valid covariance matrix, i.e. a positive-definite matrix; we say that the kernel k must be a *positive-definite function*. One often writes $f \sim \mathcal{GP}(m, k)$ to mean that $(f(t))_{t\in\mathcal{T}}$ is a Gaussian process with f.d.d.'s specified by the mean function m and kernel k.

Given some observations $y(t_1), \ldots, y(t_N)$ of an unknown function f, a Gaussian process can be used to make predictions about that function f at unseen inputs. Assume that $f \sim \mathcal{GP}(m, k)$, where the mean function m and kernel k best express our prior knowledge about the unknown function f. The observations $y(t_1), \ldots, y(t_N)$ are typically contaminated with noise. One way of modelling this is to assume that the noise $(\varepsilon(t))_{t\in\mathcal{T}}$ is identically and independently distributed with distribution $\mathcal{N}(0, \sigma^2)$. An observation $y(t)$ is then modelled—we also say *generated*—by the sum of $f(t)$ and $\varepsilon(t)$:

$$y(t) \mid f, \varepsilon = f(t) + \varepsilon(t).$$

These assumptions induce a jointly Gaussian distribution over $(y(t), f(t))_{t\in\mathcal{T}}$, another Gaussian process. We can now use the usual rules for multivariate normals to

[1] If $t_1, \ldots, t_N$ are finitely many elements from $\mathcal{T}$, then the corresponding f.d.d. is given by the distribution of the random vector $(f(t_1), \ldots, f(t_N))$.

compute the distribution of $f(t) \,|\, (y(t_1), \ldots, y(t_N))$, which is the prediction for $f(t)$ given observations $y(t_1), \ldots, y(t_N)$. This illustrates the Gaussian process modelling framework.

The kernel of a Gaussian process f usually depends on some parameters θ, also called *hyperparameters* of f. One principled way of selecting the hyperparameters θ is to use maximum likelihood estimation (Rasmussen et al., 2006):

$$\theta^* = \arg\max_{\theta} \log p(y(t_1), \ldots, y(t_N) \,|\, \theta).$$

Here $p(y(t_1), \ldots, y(t_N) \,|\, \theta)$ is also called the *(model) evidence*. Choosing θ in this way tends to explain the observations in an appropriately simple way, which is commonly recognised as a manifestation of Occam's razor (MacKay, 2002).

Gaussian processes can also be used to model multi-output functions. The construction is analogous to the single-output case: a P-dimensional multi-output Gaussian process (MOGP) is an infinite collection of real-valued random variables $(f_1(t), \ldots, f_P(t))_{t \in \mathcal{T}}$ indexed on some index set $\mathcal{T}$ such that all f.d.d.'s are Gaussian and consistent in the aforementioned sense. The mean and covariance f.d.d.'s are now specified by a *multi-output* mean function $m \colon \mathcal{T} \to \mathbb{R}^P$ and *multi-output* kernel $K \colon \mathcal{T} \to \mathbb{R}^{P \times P}$:

$$m(t) = \begin{bmatrix} \mathbb{E}(f_1(t)) \\ \vdots \\ \mathbb{E}(f_p(t)) \end{bmatrix}, \quad K(t, t') = \begin{bmatrix} \operatorname{cov}(f_1(t), f_1(t')) & \cdots & \operatorname{cov}(f_1(t), f_P(t')) \\ \vdots & \ddots & \vdots \\ \operatorname{cov}(f_P(t), f_1(t')) & \cdots & \operatorname{cov}(f_P(t), f_P(t')) \end{bmatrix}.$$

Similar to the one-dimensional case, one often collects $f(t) = \begin{bmatrix} f_1(t) & \cdots & f_P(t) \end{bmatrix}^\mathsf{T}$ and writes $f \sim \mathcal{GP}(m, K)$ to mean that $(f(t))_{t \in \mathcal{T}}$ is a multi-output Gaussian process with f.d.d.'s specified by the multi-output mean function m and multi-output kernel K.

I.2.1 Choice of Kernel

Table 1 list of five commonly used kernels and the assumptions about the function that they encode. Out these five, perhaps the most commonly used one is the exponentiated-quadratic (EQ) kernel, also known as the squared-exponential (SE) kernel, given by

$$k(t, t') = \sigma^2 \exp\left(-\frac{1}{2\ell^2} \|t - t'\|^2\right),$$

Name	Form of k	Assumption about $f \sim \mathcal{GP}(0, k)$
Constant	$k(t, t') = \sigma^2$	Function is constant
Linear	$k(t, t') = t \cdot t'$	Function is linear
Exponentiated quadratic (EQ)	$k(t, t') = \exp\left(-\frac{1}{2}\|t - t'\|^2\right)$	Function is smooth and wiggles in 1D on length scale $\sqrt{\pi/2} \approx 1.2$
Rational quadratic (RQ)	$k(t, t') = \left(1 + \frac{\|t - t'\|^2}{2\alpha}\right)^{-\alpha}$	Function is smooth and wiggles on many length scales
Exponential	$k(t, t') = \exp(-\|t - t'\|)$	Function is nondifferentiable and wiggles on length scale 1

Table 1: List of five commonly used kernels and the assumptions about the function that they encode. See Chapter 4 in the book by Rasmussen et al. (2006) for a comprehensive exposition on covariance functions.

where σ^2 is the variance and ℓ the length scale; Figure 1 shows samples from a GP with an EQ for various length scales. If the kernel is modified in the following way,

$$k(t, t') = \sigma^2 \exp\left(-\sum_{d=1}^{D} \frac{(t_d - t'_d)^2}{2{\ell_d}^2}\right),$$

then it is often said to have automatic relevance determination (ARD). What sets apart the SEARD kernel from the standard SE is that every dimension of the input now has its own length scale. This way, when the hyperparameters are learned, the kernel can scale the data independently in every dimension. Roughly, the length scale ℓ_d determines the length of a "wiggle" in dimension d, and determines how far away from the data the Gaussian process can extrapolate.

I.3 Deep Gaussian Processes

Deep neural networks (DNNs) have garnered a great deal of attention in the machine learning community due to their impressive performance in pattern recognition tasks and in applied domains such as computer vision and natural language processing (LeCun et al., 2015). Deep neural networks are large, flexible, parametric models. They have the problem that the architecture needs to be specified. And once specified,

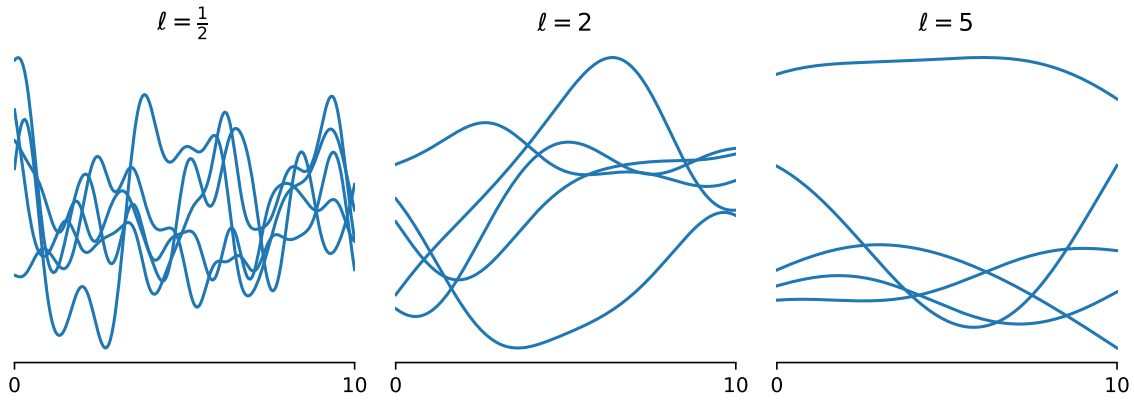


Figure 1: Samples from a GP with an EQ kernel for various length scales ℓ

the complexity of the network is fixed, regardless of the size of the data set. Deep Gaussian Processes (DGPs; Damianou (2014)) combine the representational power of DNNs with the ability of Gaussian processes to increase in complexity as more data is observed.

A Deep Gaussian Process is a distribution over functions constructed by composing functions that each have a Gaussian process prior. For example, if $f^{(1)} \sim \mathcal{GP}(0, k^{(1)})$, $\ldots$, $f^{(N)} \sim \mathcal{GP}(0, k^{(N)})$, then a Deep Gaussian Process can be constructed by

$$f(t) = (f^{(N)} \circ \cdots \circ f^{(1)})(t) = f^{(N)}(\cdots f^{(1)}(t) \cdots).$$

Performing inference in a Deep Gaussian Process model is not easy; one most resort to approximate inference techniques.

In a sense, DGPs are actually not that different from Bayesian DNNs. Matthews et al. (2018) have recently shown that, under broad conditions, the random function implied by a Bayesian neural network converges in distribution to a Gaussian process as the architecture becomes wide, an observation first made by Neal (1995).

I.4 Tools From Spectral Theory

Recall that the kernel determines the nature of the kinds of functions that can be learned by the Gaussian process. A kernel function k is called *stationary*, or *shift-invariant*, if $k(t, t') = k(t - t', 0)$. If this is the case, we often denote $\tau = t - t'$ and abuse notation to write the kernel as a function of just one variable: $k(\tau) = k(t - t', 0)$. Intuitively, if

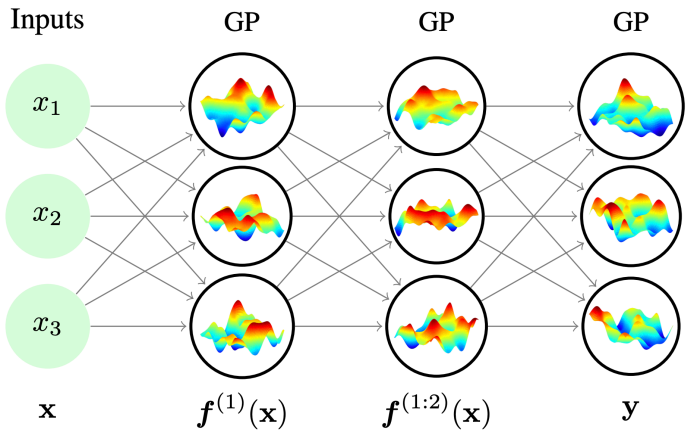


Figure 2: Visualisation of a vector-valued Deep Gaussian Process. Figure taken from D. Duvenaud et al. (2014).

the kernel is stationary, then the covariance between $f(t)$ and $f(t')$ only depends on how far t and t' are apart.

The two main tools from spectral theory that we will be using, but will not prove, are Mercer's Theorem and Bochner's Theorem:

Theorem I.1 (Mercer's)**.** Let (X, μ) be a finite measure space and $k \in L^\infty(X^2, \mu^2)$ be a kernel such that

$$T_k \colon L^2(X,\mu) \to L^2(X,\mu), \qquad T_k(f) = x \mapsto \int k(x,x')f(x')\,\mathrm{d}\mu(x')$$

is a positive-definite map. Let $(\phi_i)_{i=1}^\infty$ be the normalised eigenfunctions of T_k associated with positive eigenvalues $(\lambda_i)_{i=1}^\infty$. Then the eigenvalues are absolutely summable and

$$k(x,x') = \sum_{i=1}^{\infty} \lambda_i \phi_i(x) \phi_i^*(x')$$

holds μ^2-a.e., where the series converges absolutely and uniformly μ^2-a.e. ◀

Theorem I.2 (Bochner's)**.** An integrable function $k \colon \mathbb{R}^D \to \mathbb{R}$ is a stationary kernel and continuous at zero if and only if there exists a nonnegative function $s \colon \mathbb{R}^D \to \mathbb{C}$

such that

$$k(\tau) = \int e^{\iota \tau^\mathsf{T} \omega} s(\omega) \, \mathrm{d}\omega,$$

where ι denotes the complex unit. ◀

Bochner's Theorem admits a multivariate extension, known as Cramér's Theorem, which we also state in terms of densities (Parra et al., 2017):

Theorem I.3 (Cramér's (Cramer, 1940))**.** An integrable function $K \colon \mathbb{R}^D \to \mathbb{R}^{P \times P}$ is a stationary multi-output kernel with all $(K_{ii})_{i=1}^P$ continuous at zero if and only if there exists a nonnegative[2] function $S \colon \mathbb{R}^D \to \mathbb{C}^{P \times P}$ such that

$$K(\tau) = \int e^{\iota \tau^\mathsf{T} \omega} S(\omega) \, \mathrm{d}\omega.$$ ◀

[2] $S(\omega)$ must be a nonnegative-definite matrix for all ω.

1 Topic 1: Kernel Approximation

1.1 Random Fourier Features

Random Fourier Features (RFFs; Rahimi et al. (2008)) are a way of alleviating the $O(N^3)$ computational complexity that Gaussian processes have by mapping the data to a randomised, low-dimensional feature space and applying fast, linear methods. In this section, we show how Random Fourier Features construct an approximation of the kernel that is structured in a way such that fast, linear methods apply, as we will see in the next section.

Bochner's Theorem (Theorem I.2) tells us that every stationary kernel is characterised by its Fourier transform $s(\omega)$, a quantity that we call the power spectral density (PSD). The PSD is nonnegative and integrable, so we can associate a probability density to it:

$$p(\omega) = \frac{1}{Z} s(\omega), \quad Z = \int s(\omega)\,\mathrm{d}\omega = k(0),$$

where $Z = k(0)$ holds because because integrating the PSD yields the power of the process: $Z = \mathbb{E}[f^2(t)] = k(0)$. Then, using that the PSD is symmetric because the kernel is real valued,

$$\begin{aligned}
k(t-t') &= \int e^{\imath\omega^\mathsf{T}(t-t')} s(\omega)\,\mathrm{d}\omega, \\
&= \frac{1}{2}k(0) \int \left(e^{\imath\omega^\mathsf{T}(t-t') + e^{-\imath\omega^\mathsf{T}(t-t')}}\right) p(\omega)\,\mathrm{d}\omega, \\
&= \frac{1}{2}k(0) \int \cos(\omega^\mathsf{T}(t-t')) p(\omega)\,\mathrm{d}\omega, \\
&= \frac{1}{2}k(0)\mathbb{E}_{p(\omega)}[\cos(\omega^\mathsf{T}(t-t'))], \\
&= \frac{1}{2}k(0)\mathbb{E}_{p(\omega)}[\cos(\omega^\mathsf{T}t)\cos(\omega^\mathsf{T}t') + \sin(\omega^\mathsf{T}t)\sin(\omega^\mathsf{T}t')].
\end{aligned}$$

Therefore, by taking a Monte Carlo approximation of the expectation, we find

$$\begin{aligned}
k(t-t') &\approx \phi^\mathsf{T}(t)\phi(t'), \\
\phi(t) &= \sqrt{\frac{k(0)}{2M}} \left[\cos(\omega_1^\mathsf{T}t)\ \sin(\omega_1^\mathsf{T}t)\ \cdots\ \cos(\omega_M^\mathsf{T}t)\ \sin(\omega_M^\mathsf{T}t)\right]^\mathsf{T}
\end{aligned}$$

where all $\omega_i \sim p(\omega)$. These random features ϕ are called *Random Fourier Features*.

Sometimes an alternative estimator is presented, which can be derived in the following way. Note that

$$\int_0^{2\pi} \cos(a + 2b)db = 0$$

for any $a \in \mathbb{R}$. In particular,

$$\begin{aligned}
0 &= \mathbb{E}_{p(\omega)}\left[0 \cdot \frac{1}{2\pi}\right] \\
&= \mathbb{E}_{p(\omega)}\left[\int_0^{2\pi} \cos(\omega^\mathsf{T}(t + t') + 2b) \cdot \frac{1}{2\pi} db\right] \\
&= \mathbb{E}_{p(b)p(\omega)}[\cos(\omega^\mathsf{T}(t + t') + 2b)]
\end{aligned}$$

where $p(b)$ is the uniform distribution over $[0, 2\pi]$. Then, using the identity

$$\cos(x - y) + \cos(x + y) = 2\cos(x)\cos(y),$$

we see that

$$\begin{aligned}
k(t - t') &= \frac{1}{2}k(0)\big(\mathbb{E}_{p(\omega)p(b)}[\cos(\omega^\mathsf{T}(t - t') + b - b)] \\
&\qquad + \mathbb{E}_{p(\omega)p(b)}[\cos(\omega^\mathsf{T}(t + t') + b + b)]\big) \\
&= k(0)\mathbb{E}_{p(\omega)p(b)}[\cos(\omega^\mathsf{T}t + b)\cos(\omega^\mathsf{T}t' + b)].
\end{aligned}$$

Therefore, by taking a Monte Carlo approximation of the expectation, we find

$$\begin{aligned}
k(t - t') &\approx \phi^\mathsf{T}(t)\phi(t'), \\
\phi(t) &= \sqrt{\frac{k(0)}{M}}\big[\cos(\omega_1^\mathsf{T}t + b_1) \ \cdots \ \cos(\omega_M^\mathsf{T}t + b_M)\big]^\mathsf{T}
\end{aligned}$$

where all $\omega_i \sim p(\omega)$ and $b_i \sim p(b)$.

1.2 Random Fourier Features for Gaussian Processes

Substituting the kernel with the approximation of the kernel yielded by RFFs significantly simplifies the computation of the posterior distribution of the Gaussian process.

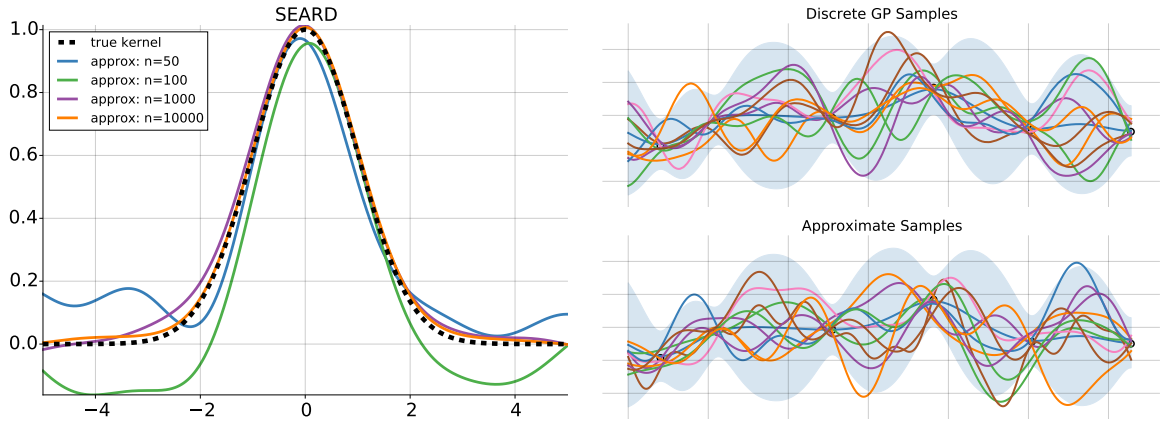


Figure 3: *Left*: approximating the SEARD kernel with 50, 100, 1000, and 10000 random features. *Right*: Exact samples ("Discrete GP Samples") and random feature samples ("Approximate Samples") from a GP predictive distribution.

Consider some observations $y = (y(t_1), \ldots, y(t_N))^\mathsf{T}$ corrupted by noise:

$$y(t) \,|\, f \sim \mathcal{N}(f(t), \sigma^2).$$

The conditioning formulas for Gaussian processes then tell us that the posterior predictive distribution is also a Gaussian process:

$$\begin{aligned} p(f \,|\, y) &= \mathcal{GP}(m, k), \\ m(t) &= \phi^\mathsf{T}(t)\Phi^\mathsf{T}(\Phi\Phi^\mathsf{T} + \sigma^2 I)^{-1} y, \\ k(t, t') &= \phi^\mathsf{T}(t)\phi(t') - \phi^\mathsf{T}(t)\Phi^\mathsf{T}(\Phi\Phi^\mathsf{T} + \sigma^2 I)^{-1}\Phi\phi(t'), \end{aligned}$$

where

$$\Phi = \begin{bmatrix} \phi(t_1) & \cdots & \phi(t_N) \end{bmatrix}^\mathsf{T}.$$

Since we have N data points, computing these expressions would take $O(N^3)$ time. However, an application of the matrix inversion lemma shows that

$$\begin{aligned} m(t) &= \phi^\mathsf{T}(t)(\Phi^\mathsf{T}\Phi + \sigma^2 I)^{-1}\Phi^\mathsf{T} y, \\ k(t, t') &= \sigma^2 \phi^\mathsf{T}(t)(\Phi^\mathsf{T}\Phi + \sigma^2 I)^{-1}\phi(t'), \end{aligned}$$

and these expressions can be computed in $O(NM^2 + M^3)$ time!

The approximate model obtained by replacing the kernel with the approximation of the kernel yielded by RFFs can be interpreted through the lens of an equivalent generative model. The first formulation of RFFs corresponds to assuming that the underlying function f is modelled by

$$f(t) = \phi^{\mathsf{T}}(t)\theta = \sqrt{\frac{k(0)}{2M}}\left[\sum_{i=1}^{M} Z_i^{(1)} \cos(\omega_i t) + \sum_{i=1}^{M} Z_i^{(2)} \sin(\omega_i t)\right]$$

where $(Z_i^{(1)}, Z_i^{(2)})_{i=1}^M$ are i.d.d. $\mathcal{N}(0,1)$ variables, all concatenated into a random vector θ. Similarly, the second formulation of RFFs corresponds to assuming that the underlying function f is modelled by

$$f(t) = \phi^{\mathsf{T}}(t)\theta = \sqrt{\frac{k(0)}{M}} \sum_{i=1}^{M} Z_i \cos(\omega_i t + b_i)$$

where $(Z_i)_{i=1}^M$ are i.d.d. $\mathcal{N}(0,1)$ variables, again all concatenated into a random vector θ.

Although RFFs construct an unbiased estimate of the kernel, it is not guaranteed that the sampled parameters $(\omega_i)_{i=1}^M$ (first formulation) or $(\omega_i, b_i)_{i=1}^M$ (second formulation) fit the data best. Therefore, after sampling, one could consider to optimise these parameters through optimising the model evidence. This gives a better fit to the data, but the approximation of the kernel is no longer unbiased. Sampling RFFs and then optimising the sampled parameters is known as the Sparse Spectrum Approximation (SSA) for Gaussian processes (Lázaro-Gredilla et al., 2010).

1.3 Random Fourier Features for Deep Gaussian Processes

As mentioned in Section I.3, inference in Deep Gaussian Processes (DGPs) is difficult, so approximate methods are often used. Cutajar et al. (2017) propose to use RFFs to approximate a DGP with the goal of making inference easier.

Let

$$f(t) = (f^{(N)} \circ \cdots \circ f^{(1)})(t)$$

be a DGP, where each $f^{(i)} \sim \mathcal{GP}(0, K^{(i)})$ is a *vector-valued* Gaussian process. By using RFFs to approximate the kernel of every $f^{(i)}$, we obtain approximations of the

GPs:

$$f^{(i)}(t) \approx \hat{f}^{(i)}(t) = \Phi^{(i)\mathsf{T}}(t)\Theta^{(i)}$$

where $\Phi^{(i)}$ denotes the RFFs for $K^{(i)}$ and Θ is a random matrix with $\mathcal{N}(0,1)$ elements. Substituting these approximate models into the expression for the DGP, we get an approximation of the DGP, called RFF-DGP:

$$\begin{aligned} f(t) &\approx \hat{f}(t) \\ &= (\hat{f}^{(N)} \circ \cdots \circ \hat{f}^{(1)})(t) \\ &= \Phi^{(N)\mathsf{T}}(\Phi^{(N-1)\mathsf{T}}(\cdots \Phi^{(1)\mathsf{T}}(t)\Theta^{(1)} \cdots)\Theta^{(N-1)})\Theta^{(N)}. \end{aligned}$$

This reveals an exciting connection between the approximate DGP and Bayesian neural nets: $\hat{f}$ is a Bayesian neural net with a specific architecture and independent $\mathcal{N}(0,1)$ priors on the weights!

Cutajar et al. (2017) go on to describe architectures corresponding to specific kernels. For example, by a specific choice of kernels, ReLU activations can be recovered. We refer the reader to their paper for further details.

1.3.1 Inference in RFF-DGP

Let Φ be the collection of all kernel parameters, W be the collection of all angular frequencies sampled in the RFF approximations, and $\Theta = (\Theta^{(1)}, \ldots, \Theta^{(N)})$ be the collection of all random matrices. In performing inference in the RFF-DGP approximation, for each of these quantities we can either attempt to compute a posterior distribution or simply use a point estimate; RFF-DGP specifices a prior for W and Θ, but a prior over Φ remains to be chosen. For parameters that we choose to perform posterior inference for, Cutajar et al. (2017) propose to use variational inference as the approximate inference scheme, a common practice in Bayesian neural networks (Blundell et al., 2015).

Suppose, for the purpose of illustration, that we would like to compute point estimates for Φ and Θ, but compute an approximate posterior for W. Let Y be some data set. The both the posterior $p(W \mid Y, \Phi, \Theta)$ and the log marginal likelihood $\log p(Y \mid \Phi, \Theta)$ involve intractable integrals, so they cannot be evaluated. Variational inference gets around this by instead maximising a lower bound on the model evidence called the

evidence lower bound (ELBO):

$$\log p(Y \mid X, \Phi, \Theta) \geq \mathbb{E}_{q(W)}[\log p(Y \mid X, W, \Phi, \Theta)] - \mathrm{D}_{\mathrm{KL}}(q(W) \,\|\, p(W))$$

where $q(W)$ is an approximation of the posterior $p(W \mid Y, \Phi, \Theta)$. We are free to select any parametrisation we like for $q(W)$, with various performance trade-offs; a commmon choice is a factorised Gaussian.

The evidence lower bound can be maximised with gradient-based optimisation. In this case that $q(W)$ is a factorised Gaussian, we can compute $\mathrm{D}_{\mathrm{KL}}(q(W) \,\|\, p(W))$ analytically and thus its gradients automatically using automatic differentiation, e.g. with the Python libraries PyTorch or TensorFlow. However, $\mathbb{E}_{q(W)}[\log p(Y \mid X, W, \Phi, \Theta)]$ cannot be computed in closed-form. We get around this by using a Monte Carlo approximation, sampling $W \sim q(W)$. Importantly, since $q(W)$ contains parameters that need to be optimised and W is sampled from $q(W)$, we need to be careful in computing gradients of the Monte Carlo approximation. A common solution is to use the *reparametrisation trick* (Kingma et al., 2013): we "reparametrise" samples from $q(w) = \mathcal{N}(\mu, \sigma^2)$ by, instead of sampling from $q(w)$ directly, we sample $\varepsilon \sim \mathcal{N}(0, 1)$ and then set $w = \sigma\varepsilon + \mu$. The reparametrisation trick ensures that randomness in the Monte Carlo approximation of the expectation is fixed when when computing gradients with respect to the parameters of $q(W)$, resulting in a lower-variance, unbiased estimator of the gradient of the ELBO.

2 Topic 2: Kernel Design

Random Fourier Features (RFFs) alleviate the first major issue with Gaussian processes, which is their cubic scaling in the number of data points n. RFFs do not, however, help with the choice of kernel, a problem that we call *kernel design*. A kernel can be designed by coming up with a parametrisation that encodes one's assumptions about the underlying function. This is not always easy, because it can be unclear how these assumptions may be encoded into a kernel, and the kernel must be a positive-definite function.

In this topic, we will see how Bochner's Theorem (Theorem I.2) can be used to design flexible kernels whose parameters may be learned from data. Bochner's Theorem (Theorem I.2) says that a stationary kernel is characterised by its power spectral density (PSD), which is the distribution of power contained in frequency components that make up the signal. The PSD needs to be nonnegative and symmetric (in the case of a real-valued function), which are simpler requirements than positive definiteness. Therefore, instead of attempting to flexibly parametrise a kernel directly, one could flexibly parametrise the PSD. We will discuss a number of techniques that take this approach.

2.1 The Spectral Mixture Kernel

We have already seen that RFFs approximate the PSD by a carefully sampled line spectrum, which are in the Sparse Spectrum Approximation (SSA) subsequently optimised. Assuming a line spectrum is equivalent to assuming that the signal consists of a sum of finitely many sines, which is a strong parametric assumption. Wilson et al. (2013) propose to fatten these lines and instead parametrise the PSD with a symmetric Gaussian mixture:

$$s(\omega) = \frac{1}{2}\sum_{q=1}^{Q} w^{(q)}\big(\mathcal{N}\big(\omega; \mu^{(q)}, \Sigma^{(q)}\big) + \mathcal{N}\big(\omega; -\mu^{(q)}, \Sigma^{(q)}\big)\big).$$

Then

$$k^{(\mathrm{SMK})}(\tau) = \sum_{q=1}^{Q} w^{(q)} \exp\big(-\tfrac{1}{2}\tau^{\mathsf{T}}\Sigma^{(q)}\tau\big)\cos\big(\mu^{(q)\mathsf{T}}\tau\big),$$

which Wilson et al. (2013) call the Spectral Mixture Kernel (SMK). Here the weights $(w^{(q)})_{q=1}^{Q}$ specify the contributions of the mixture components, $(\mu^{(q)})_{q=1}^{Q}$ the frequencies, and $(\Sigma^{(q)})_{q=1}^{Q}$ the squared inverse length scales, which determine how quickly a component varies with the input.

The SMK can be interpreted in terms of a generative model, which is a truncated Fourier series with time-varying coefficients:

$$\begin{aligned} f^{(\mathrm{SMK})}(t) &= \sum_{q=1}^{Q} \sqrt{w^{(q)}}(c_1^{(q)}(t)\cos(\mu^{(q)\mathsf{T}}t) + c_2^{(q)}(t)\sin(\mu^{(q)\mathsf{T}}t)) \\ &\sim \mathcal{GP}\big(0, k^{(\mathrm{SMK})}(\tau)\big) \end{aligned}$$

where $c_1^{(q)}, c_2^{(q)} \sim \mathcal{GP}(0, \exp(-\frac{1}{2}\tau^{\mathsf{T}}\Sigma^{(q)}\tau))$ are coefficients that vary on inverse length scales $(\Sigma^{(q)})_{q=1}^{Q}$. This shows that, compared to the SSA where $(c_1^{(q)}, c_2^{(q)})_{q=1}^{Q}$ are constant, the spectral lines are fattened to Gaussians by allowing the coefficients to vary with time.

The SMK can be used as a flexible, drop-in replacement for popular kernels that retains simple training and inference procedures. Even with a small number of components, $Q \leq 10$, the SMK is able to closely recover many standard kernels (Wilson et al., 2013). Additionally, the SMK is able to express negative covariances, which is key in modelling e.g. linear trends. Two downsides of the SMK are that (1) it is usually not clear in advance how many components are needed and (2) the hyperparameters $(w^{(q)}, \mu^{(q)}, \Sigma^{(q)})_{q=1}^{Q}$ are sensitive to initialisation and hard to optimise, leading to a nonconvex optimisation problem with many local minima.

2.1.1 Multiple Outputs

Parra et al. (2017) extend the SMK to multiple outputs using the multivariate extension of Bochner's Theorem (Theorem I.2), Cramér's Theorem (Theorem I.3). They call their generalisation the Multi-Output Mixture Kernel (MOSMK). The construction is analogous to the single-output case, but extra care must be taken to satisify non-negativity of the density. The MOSMK models the PSD with a symmetrix mixture of

outer products of Gaussian vectors:

$$\begin{aligned}
S(\omega) &= \frac{1}{2}\sum_{q=1}^{Q}\big(R^{(q)}(\omega)R^{(q)\dagger}(\omega) + R^{(q)}(-\omega)R^{(q)\dagger}(-\omega)\big),\\
R_i^{(q)}(\omega) &= w_i^{(q)}\exp\Big(-\tfrac{1}{4}(\omega-\mu_i^{(q)})\Sigma_i^{(q)-1}(\omega-\mu_i^{(q)}) - \iota(\theta_i^{(q)\mathsf{T}}\omega + \phi_i^{(q)})\Big)
\end{aligned}$$

where S is now a $(P\times P)$-matrix-valued function and $\cdot^\dagger$ denotes conjugate transposition and where every component further includes *time delays* $(\theta_i^{(q)})_{i=1}^P$ and *phase shifts* $(\phi_i^{(q)})_{i=1}^P$. Then

$$K_{ij}(\tau) = \sum_{q=1}^{Q}\alpha_{ij}^{(q)}\exp\Big(-\tfrac{1}{2}(\tau+\theta_{ij}^{(q)})^{\mathsf{T}}\Sigma_{ij}^{(q)}(\tau+\theta_{ij}^{(q)})\Big)\cos\Big((\tau+\theta_{ij}^{(q)})^{\mathsf{T}}\mu_{ij}^{(q)} + \phi_{ij}^{(q)}\Big)$$

where

$$\begin{aligned}
\alpha_{ij}^{(q)} &= w_i^{(q)}w_j^{(q)}(2\pi)^{\frac{d}{2}}|\Sigma_{ij}^{(q)}|^{\frac{1}{2}}\exp\Big(-\tfrac{1}{4}(\mu_i^{(q)}-\mu_j^{(q)})^{\mathsf{T}}(\Sigma_i^{(q)}+\Sigma_j^{(q)})^{-1}(\mu_i^{(q)}-\mu_j^{(q)})\Big),\\
\Sigma_{ij}^{(q)} &= 2\Sigma_i^{(q)}(\Sigma_i^{(q)}+\Sigma_j^{(q)})^{-1}\Sigma_j^{(q)},\\
\mu_{ij}^{(q)} &= (\Sigma_i^{(q)}+\Sigma_j^{(q)})^{-1}(\Sigma_i^{(q)}\mu_j^{(q)} + \Sigma_j^{(q)}\mu_i^{(q)}),\\
\theta_{ij}^{(q)} &= \theta_i^{(q)} - \theta_j^{(q)},\\
\phi_{ij}^{(q)} &= \phi_i^{(q)} - \phi_j^{(q)}.
\end{aligned}$$

The MOSMK can also be interpreted in terms of a generative model, which is for each output a truncated Fourier series with time-varying coefficients, also including time delays and phase shifts:

$$\begin{aligned}
f_i^{(\mathrm{MOSMK})}(t) &= \sum_{q=1}^{Q} w_i^{(q)}\Big(c_{i1}^{(q)}(t-\theta_i^{(q)})\cos\Big(\mu_i^{(q)\mathsf{T}}(t-\theta_i^{(q)}) + \phi_i^{(q)}\Big)\\
&\qquad\qquad + c_{i2}^{(q)}(t-\theta_i^{(q)})\sin\Big(\mu_i^{(q)\mathsf{T}}(t-\theta_i^{(q)}) + \phi_i^{(q)}\Big)\Big)\\
&\sim \mathcal{GP}\big(0, k^{(\mathrm{MOSMK})}(\tau)\big)
\end{aligned}$$

where $(c_{i1}^{(q)})_{i=1}^P$ and $(c_{i2}^{(q)})_{i=1}^P$ are independent multi-output Gaussian processes with kernels

$$\mathbb{E}[c_{ik}^{(q)}(t)c_{jk}^{(q)}(t')] = \frac{\alpha_{ij}^{(q)}}{w_i^{(q)}w_j^{(q)}}\exp\Big(-\tfrac{1}{2}(t-t')^{\mathsf{T}}\Sigma_{ij}^{(q)}(t-t')\Big).$$

Other approaches to extending the SMK to multiple outputs involve that by Ulrich et al. (2015) and Chen et al. (2018).

2.1.2 Nonstationary Signals

Before extending the SMK to nonstationary signals, let us first consider the simple exponentiated-quadratic (EQ) kernel. Even extending the EQ kernel to nonstationary signals is not straightforward: if the length scales are made input dependent, then the resulting function is not positive definite. Gibbs (1997) demonstrates how such an extension can instead be done, resulting in a nonstationary version of the EQ kernel called the *Gibbs kernel*:

$$k^{(\text{Gibbs})}(t, t') = \prod_{d=1}^{D} \sqrt{\frac{2\ell_d(t)\ell_d(t')}{\ell_d^2(t) + \ell_d^2(t')}} \exp\left(-\sum_{d=1}^{D} \frac{(t_d - t'_d)^2}{\ell_d^2(t) + \ell_d^2(t')}\right)$$

where $(\ell_d)_{d=1}^D$ are positive functions parametrising the length scales of the input dimensions. The important part of the Gibbs kernel is the prefactor of the exponential, which may seem odd at first. It is there to ensure positive definiteness and engineered in a way such that $k^{(\text{Gibbs})}(t, t) = 1$.

Let us detail this construction by Gibbs in one dimension. The EQ kernel can be derived by considering basis functions $\phi(t; c)$ centred around c with length scale ℓ,

$$\phi(t; c) = \left(\sqrt{\frac{2}{\pi}}\frac{1}{\ell}\right)^{\frac{1}{2}} \exp\left(-\frac{1}{\ell^2}(t - c)^2\right),$$

and summing them against a white noise process $n(t) \sim \mathcal{GP}(0, \delta(t - t'))$:

$$f(t) \mid n = \int_{-\infty}^{\infty} \phi(t; c) n(c) \, \mathrm{d}c.$$

Then, after some algebra, $f \sim \mathcal{GP}(0, k)$ with

$$k(t, t') = \int_{-\infty}^{\infty} \phi(t; c)\phi(t'; c) \, \mathrm{d}c = \exp\left(-\frac{1}{2\ell^2}(t - t')^2\right),$$

which is the EQ kernel. Gibbs proposes to make the length scales of the basis functions

dependent on t—*not* c, because then the integration would become intractable:

$$\phi(t;c) = \left(\sqrt{\frac{2}{\pi}}\frac{1}{\ell(t)}\right)^{\frac{1}{2}} \exp\left(-\frac{1}{\ell^2(t)}(t-c)^2\right).$$

Then

$$k(t,t') = \int_{-\infty}^{\infty} \phi(t;c)\phi(t';c)\,\mathrm{d}c = \sqrt{\frac{2\ell(t)\ell(t')}{\ell^2(t)+\ell^2(t')}} \exp\left(-\frac{(t-t')^2}{\ell^2(t)+\ell^2(t')}\right),$$

which is the Gibbs kernel in one dimension.

Based on the Gibbs kernel, Remes et al. (2017) propose a nonstationary generalisation of the SMK, named the Generalised Spectral Mixture Kernel (GSMK):

$$k^{(\mathrm{GSMK})}(t,t') = \sum_{q=1}^{Q} w^{(q)}(t)w^{(q)}(t')k_q^{(\mathrm{Gibbs})}(t,t')\cos\big(\mu^{(q)\mathsf{T}}(t)t - \mu^{(q)\mathsf{T}}(t')t'\big),$$

where $(w^{(q)})_{q=1}^{Q}$ and $(\mu^{(q)})_{q=1}^{Q}$ are positive functions parametrising respectively the weights and frequencies of the components. The unknown functions $(w^{(q)}, \ell^{(q)}\mu^{(q)})_{q=1}^{Q}$ are given a log-Gaussian process prior and learned with maximum a posteriori (MAP) estimation.

This generalisation can also be interpreted in terms of a generative model:

$$\begin{aligned} f^{(\mathrm{GSMK})}(t) &= \sum_{i=1}^{Q} w^{(q)}(t)\Big(c_1^{(q)}(t)\cos(\mu^{(q)\mathsf{T}}(t)t) + c_2^{(q)}(t)\sin(\mu^{(q)\mathsf{T}}(t)t)\Big) \\ &\sim \mathcal{GP}\big(0, k^{(\mathrm{GSMK})}(\tau)\big) \end{aligned}$$

where $c_1^{(q)}, c_2^{(q)} \sim \mathcal{GP}(0, k_q^{(\mathrm{Gibbs})})$ are coefficients that vary on time-varying length scales.

2.2 The Gaussian Process Convolution Model

The SMK and its extensions assume a *parametric* model for the PSD. A more flexible alternative is to model the PSD *nonparametrically*, for example by letting

$$s(\omega) = |\hat{h}(\omega)|^2$$

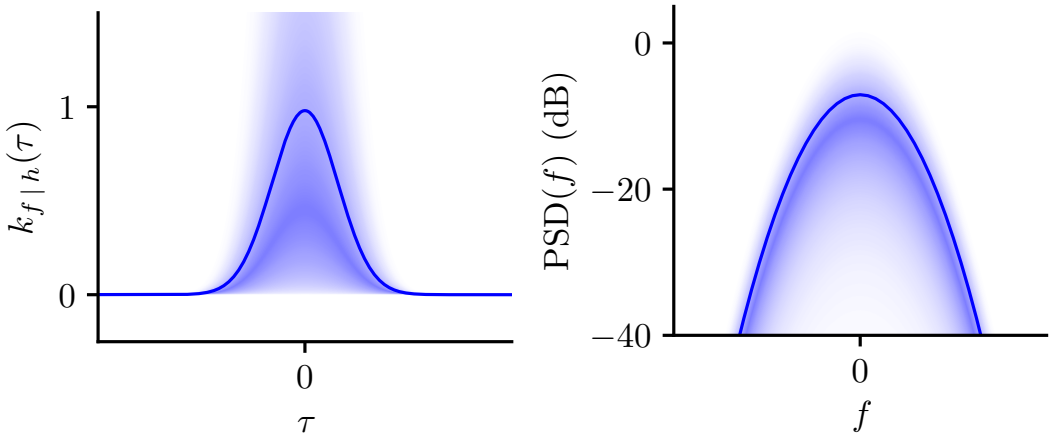


Figure 4: Prior over kernels and power spectral densities (PSDs) induced by the Gaussian Process Convolution Model (GPCM). Lines correspond to means and gradients indicate marginal variance. Figure taken from Bruinsma (2016).

where $\hat{h}\colon \mathbb{R}^d \to \mathbb{C}$ is a symmetric unknown function. Then the inverse Fourier transform h of $\hat{h}$ is a real-valued function on $\mathbb{R}^d$ and, by the Convolution Theorem,

$$k(t, t') = \int_{-\infty}^{\infty} h(t - z)h(t' - z)\,\mathrm{d}z = h * Rh(t - t')$$

where R is the reversal operator: $Rh(t) = h(-t)$. By interpreting k as an infinite-dimensional matrix, k can be seen as an infinite sum of outer products, which is positive definite. Alternatively, interpret convolution as "multiplication for functions" and reversal as "transposition for functions". Then, roughly, $k = hh^{\mathsf{T}}$; compare this to AA^{T}, which is a positive-definite matrix for any matrix A.

Tobar et al. (2015) propose to model h also with a Gaussian process, $h \sim \mathcal{GP}(0, k_h)$, resulting in a model called the Gaussian Process Convolution Model (GPCM). Care must be taken in choosing h: for the variance of the modelled signal to be finite, h must be square integrable in expectation; that is,

$$\int_{-\infty}^{\infty} k_h(t, t)\,\mathrm{d}t < \infty,$$

which can be interpreted as the requirement that k_h must have a finite trace. The GPCM induces a nonparametric prior distribution over PSDs, or equivalently over kernels; this prior is visualised in Figure 4.

The GPCM admits an equivalent interpretation in terms of linear systems. Since $s(\omega) = 1 \cdot |\hat{h}(\omega)|^2$ and white noise has a constant PSD of 1, the GPCM is equivalent to passing white noise through a linear system with impulse response h, where the impulse response h is modelled nonparametrically with another Gaussian process.

2.3 Conclusion

Instead of designing a flexible kernel directly, one can design the PSD, whose requirements are simpler than that of the kernel. Parametric approaches parametrise the PSD with a line spectrum or a Gaussian mixture, but other choices are possible. One can even use a nonparametric model for the PSD, but then inference becomes more involved.

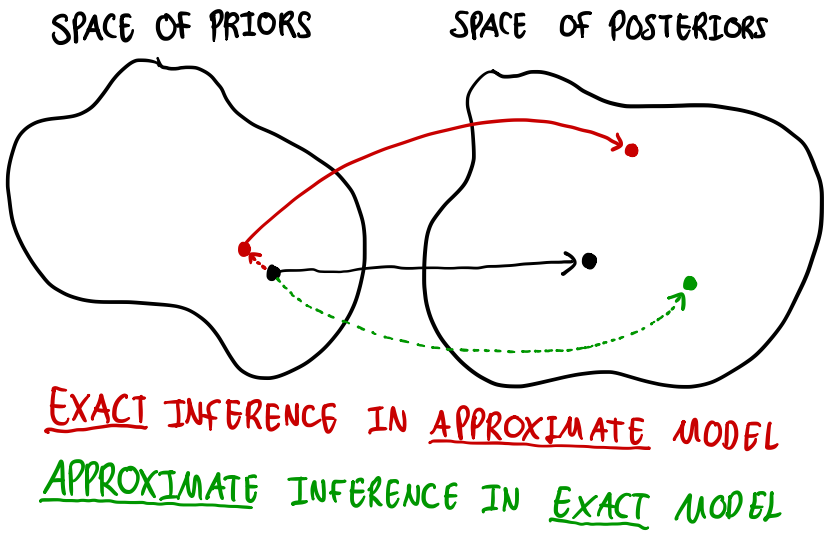


Figure 5: Exact inference in an approximate model versus approximate inference in an exact model. If you change your model prior, this might affect your posterior in unforeseen or undesired ways. A line from the space of priors to the space of posteriors represents an application of Bayes' rule. Dashed lines represent approximations.

3 Topic 3: Variational Inference

Random Fourier Features (RFFs) scale Gaussian processes to large numbers of data points N by approximating the kernel, thus forming an approximate model, and performing exact inference in this approximate model. But performing *exact* inference in an *approximate* model is a dangerous thing to do, because changing your assumptions (i.e. approximating your model) can affect your posterior in unforeseen or undesired ways (Figure 5). An approximation of the model also often introduces more parameters which need to be optimised. This comes with the risk of overfitting, as we have observed in the Sparse Spectrum Approximation (SSA). Intead, one should perform *approximate* inference in the *exact* model. There the approximation may be optimised as much as you like, without any risk of overfitting: you will only get closer to the true answer.

In this topic, we consider a second technique to scale Gaussian processes, one that aims to approximate the exact model. We will see how this second technique can be combined with RFFs to get the best of both worlds.

3.1 Inducing Points

The story told next is accompanied by the cartoon depicted in Figure 6.

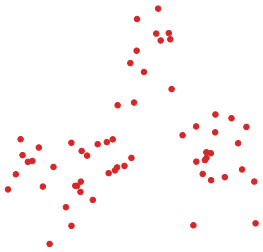
A straightforward way to scale Gaussian processes to large data sets is to only use a subset of $M \ll N$ data points, e.g. chosen by random sampling (Figures 6a and 6b). This is obviously very wasteful, but it may suffice if the predictions are accurate enough. Instead of random sampling, we can be a little bit more clever and carefully choose these M data points. For example, if the data were to consist of clusters, we could choose one representative of every cluster (Figure 6c). More generally, the subset of M data points could be chosen such that they *summarise nearby data points*. And these M data points do not have to be actual observations. Perhaps we can cleverly make up *pseudo-observations* that best summarise nearby actual data points (Figure 6d).

Pseudo-observations that summarise nearby actual data points is still not an entirely satisfactory solution, because the model will not be aware that its pseudo-observations are not actual observations. Instead, it will think that we actually observed these pseudo-observations. Therefore, if you ask it to predict at the location of a pseudo-observation, it will confidently predict you the value of that pseudo-observation, which might be far from what is actually observed (Figure 6e).

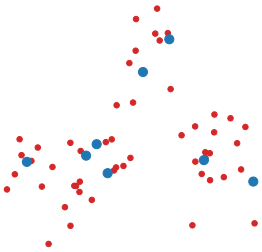
What goes wrong is that reducing multiple data points to a single one comes with a loss of *uncertainty*. If the data points summarised by a pseudo-observation have similar values, then the we can be fairly certain that the value of the pseudo-observation is close to what is actually observed. But if the summarised data points have different values, then we should be *uncertain* about the value of that pseudo-observation.

To turn the concept of pseudo-observations into a satisfactory solution, we explicitly represent this uncertainty. We do not let a pseudo-observation have a single value y, but instead assume a *distribution* over y (Figure 6f). This distribution represents the distribution of values of nearby data points that it the pseudo-observation summarises. Thus, the pseudo-observations are now *stochastic*. Stochastic pseudo-observations are better known as *inducing points* in the Gaussian process literature.[3] The effect of assuming a distribution over the values of the pseudo-observations is visualised in Figures 6g and 6h: a corrective variance arsises, which is added to the predictive

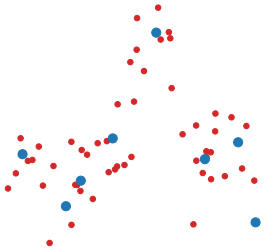
[3] The term "pseudo-observations" and the distinction between pseudo-observations and inducing points are not conventional; they are introduced here just to explain the concepts.



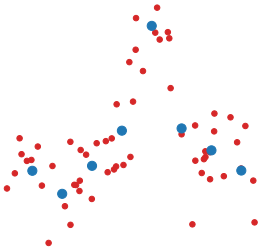
(a) Consider some random data set.



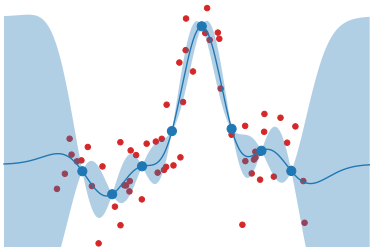
(b) We could consider a random subset, but that is very wasteful.



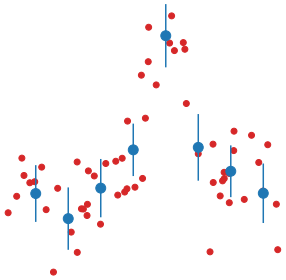
(c) We could also choose the subset a bit more clever, for example by clustering.



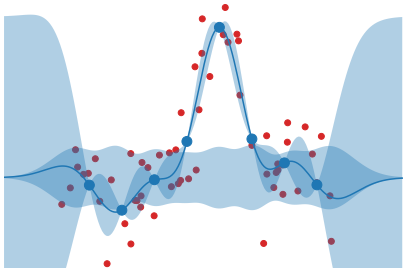
(d) Best is to introduce carefully constructed pseudo-observations.



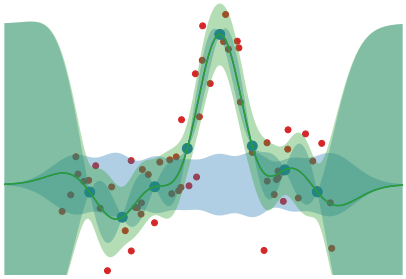
(e) The prediction looks overconfident, because pseudo-observations are considered actual observations.



(f) Introduce a distribution for each pseudo-observation. Now we have inducing points.



(g) This gives rise to a corrective variance that should appropriately inflate the prediction.



(h) When summed together, the prediction looks much more reasonable!

Figure 6: Cartoon that accompanies the story told in Section 3.1.

distribution resulting from the pseudo-observations to appropriately inflate the variance and capture most of the data. We now proceed to formalise the notions of pseudo-observations and inducing points.

Let $f \sim \mathcal{GP}(0, k)$ and let $\mathcal{D}$ be some data set of size N. The posterior of interest is $p(f \,|\, \mathcal{D})$, but computing $p(f \,|\, \mathcal{D})$ directly may be too expensive. The simplest solution is to simply use a subset of the data, corresponding to the approximation $p(f \,|\, \mathcal{D}) \approx p(f \,|\, \mathcal{D}_{\text{subset}})$. But, as we already argued, we can do better. Introduce $M \ll N$ pseudo-observations u at input locations $(t_{u,1}, \ldots, t_{u,M})$: $u = (f(t_{u,1}), \ldots, f(t_{u,M}))$. Then, given carefully chosen values for the pseudo-observations u, we could approximate $p(f \,|\, \mathcal{D}) \approx p(f \,|\, u)$. This, however, fails to take into account that the pseudo-observations are not actual observations. Therefore, turning u into inducing points, instead assume a carefully chosen distribution $q(u)$. This gives rise to the famous inducing point approximation of Gaussian processes originally proposed by Titsias (2009), a major milestone in the literature of Gaussian process approximations:

$$p(f \,|\, \mathcal{D}) \approx \int p(f \,|\, u) q(u) \,\mathrm{d}u \equiv q(f),$$

where $q(f)$ denotes the *approximate posterior* for f. Titsias (2009) proposes to choose $q(u)$ by minimising the Kullback–Leibler (KL) divergence between $q(f)$ and $p(f \,|\, \mathcal{D})$:

$$q^*(u) = \operatorname*{arg\,min}_{q(u)} \mathrm{D}_{\mathrm{KL}}(q(f) \,\|\, p(f \,|\, \mathcal{D})),$$

where

$$\mathrm{D}_{\mathrm{KL}}(q(x) \,\|\, p(x)) \equiv \int q(x) \log \frac{q(x)}{p(x)} \,\mathrm{d}x.$$

That is, the *approximate* posterior is chosen such that it best approximates the *exact* posterior, as promised in the introduction. This means that we cannot overfit due to having too many inducing points: more inducing points will only get us closer to the true posterior. The procedure of minimising the KL divergence between an approximate posterior and the true posterior is called *variational inference*.

A simple exercise in the calculus of variations shows that

$$q^*(u) \propto p(u) \exp \int p(f \mid u) \log p(\mathcal{D} \mid f) \, \mathrm{d}f.$$

Substituting $q^*(u)$ back into the expression for $q(f)$ then yields that $q(f) = \mathcal{GP}(m_q, k_q)$ where

$$\begin{aligned} m_q(t) &= k(t, t_u)(\sigma^2 K_{uu} + K_{uf} K_{fu})^{-1} K_u^{-1} K_{uf} y, \\ k_q(t, t') &= \underbrace{k(t, t') - k(t, t_u^\mathsf{T}) K_{uu}^{-1} k(t_u, t')}_{\text{posterior variance of pseudo-observations}} + \underbrace{k(t, t_u^\mathsf{T})(K_{uu} + \sigma^{-2} K_{uf} K_{fu})^{-1} k(t_u, t')}_{\text{corrective variance}}, \end{aligned}$$

where $(K_{uu})_{ij} = k(t_{u,i}, t_{u,j})$, $(K_{uf})_{ij} = k(t_{u,i}, t_j)$, y are the actual observations, and σ^2 is the observation noise. Observe that the approximate posterior variance can be decomposed into a sum of (1) the posterior variance that arises from non-stochastic pseudo-observations and (2) a corrective term deriving from that the inducing points summarise multiple observations; it is this decomposition that is shown in Figures 6g and 6h. The approximate posterior $q(f)$ can be computed in $O(NM^2)$ time, which is a significant saving compared to the $O(N^3)$ time required to compute $p(f \mid \mathcal{D})$ directly. One should verify that if an inducing point is placed at every observed data point, then the approximation becomes exact.

The inducing point approximation for Gaussian processes has been put on a rigorous grounding by G. Matthews et al. (2015) and enjoys great empirical performance (T. D. Bui et al., 2016a).

3.2 Variational Fourier Features

Inducing points for Gaussian processes can be thought of as a *spatially local* summarisation of the data. This means that inducing points need to cover the input space where the data lives and where we wish to make predictions. For a time series, regularly spaced inducing points are known to work well (Thang D. Bui et al., 2014). This reveals a deficiency of the inducing points method: as N grows and the time series becomes longer, more inducing points are needed as well; that is, $M = CN$ where the constant C may be small. The inducing points approximation therefore effectively still scales cubically, $O(NM^2) = C^2 O(N^3)$, but with a much improved constant. For very large time series, this can be problematic.

This issue is less severe for the Sparse Spectrum Approximation (SSA), because its predictive features are sines and cosines. Instead of contributing to the prediction only locally near an inducing point, these sines and cosines contribute to the prediction everywhere. Whereas inducing points summarise the data *spatially locally*, the SSA is *spectrally local* approximation of the Gaussian process.

But the SSA suffers from another issue, one that is also problematic: overfitting. This is no issue for inducing points: more inducing points will only get you closer to the desired posterior. Is it possible to combine the appealing approximative construction of the inducing points method with the representative power of Random Fourier Features (RFFs) that the SSA uses, to get rid of the hidden $O(N^3)$ scaling and to enable us to use as many frequency components as we want without running the risk of overfitting? A method called Variational Fourier Features (VFFs) (Hensman et al., 2016) aims to do exactly this.

The posterior predictive mean of the inducing point method is of the form

$$\hat{f}^{(\mathrm{IP})}(t) = \sum_{i=1}^{M} \alpha_i k(t, t_{u,i}),$$

whereas that of the SSA is of the form of a truncated Fourier series:

$$\hat{f}^{(\mathrm{SSA})}(t) = \sum_{i=1}^{M} \alpha_i \cos(2\pi\xi_i t) + \sum_{i=1}^{M} \beta_i \sin(2\pi\xi_i t).$$

VFFs is a modification of the inducing point method, done in clever way such that the posterior predictive mean $\hat{f}^{(\mathrm{VFF})}(t)$ is also of the form of a truncated Fourier series, thereby obtaining the representative power that RFFs show. A key ingredient in the construction is an extension of the inducing point method called *inter-domain inducing points* (Gredilla et al., 2009), which we will outline first.

Instead of introducing pseudo-observations for the underlying function f, we could first linearly transform f with linear transform h called an *inter-domain transformation*,

$$g(\xi) \mid f = \int_{-\infty}^{\infty} h(\xi, \tau) f(\tau) \, \mathrm{d}\tau,$$

and introduce pseudo-observations for g instead. In some cases, this construction is

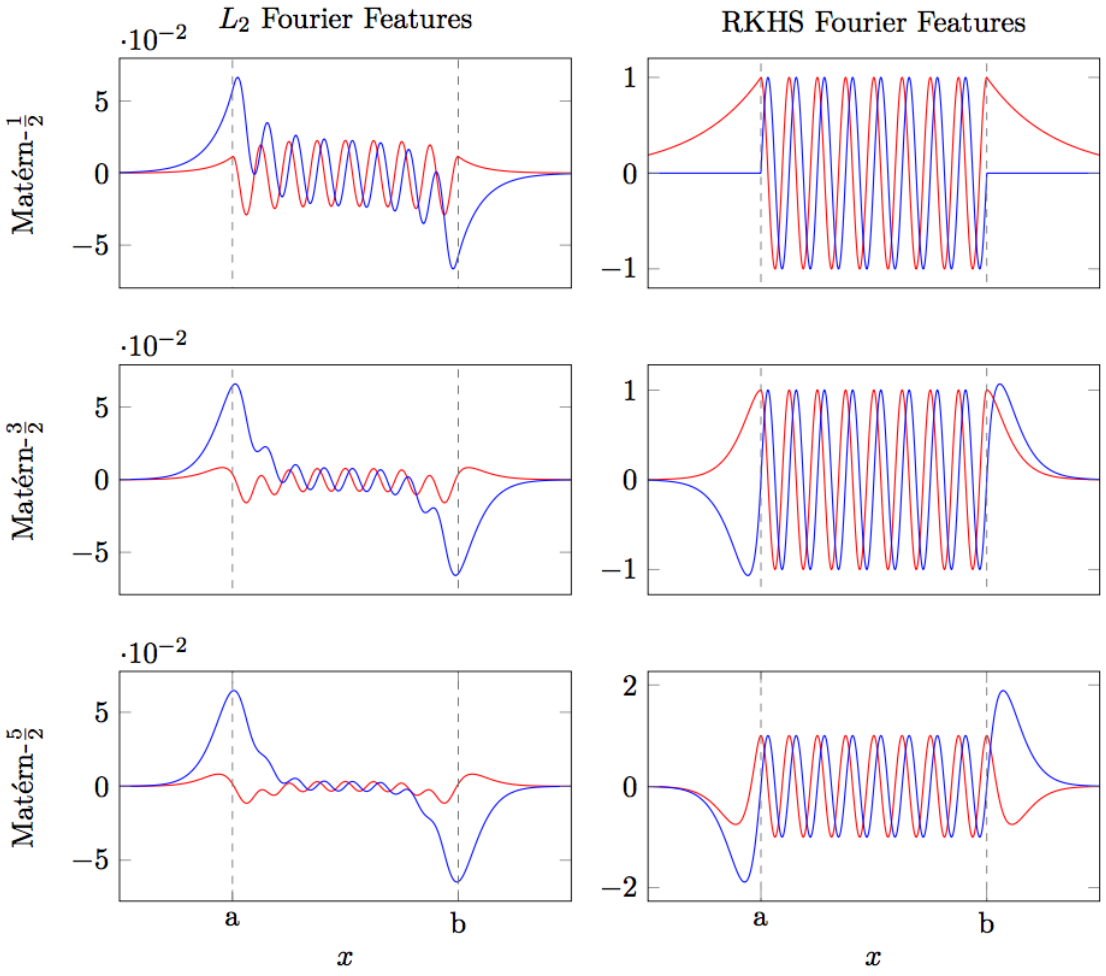


Figure 7: Predictive features from $L^2[a, b]$-VFFs (left) and VFFs derived from the RKHS construction (right) for several choices of the kernel k. Figure taken from Hensman et al. (2016).

necessary: If a priori f is assumed to be white noise, then pseudo-observations of f are unable to induce a posterior over f, because white noise is uncorrelated. But, if $h(\xi, \tau)$ is a Gaussian bump centred at ξ, then a pseudo-observation of $g(\xi)$ will correlate with $f(t)$ for all t near ξ, thus inducing a nontrivial posterior. In other cases, the inter-domain transformation may improve the approximate posterior: h can be designed to filter out irrelevant features of f and emphasise important ones. Inducing points for g are called *inter-domain inducing points*, because g "lives in a different domain" than f.

The posterior predictive mean of the inter-domain inducing point method is of the

form

$$\hat{f}^{(\mathrm{IDIP})}(t) = \sum_{i=1}^{M} \alpha_i \int_{-\infty}^{\infty} k(t,\tau) h(\xi_{u,i}, \tau)\,\mathrm{d}\tau,$$

where the inducing points for g are at $\xi_u = (\xi_{u,1}, \ldots, \xi_{u,M})$. The construction of VFFs is to engineer h in such a way that $\hat{f}^{(\mathrm{IDIP})}$ also becomes a Fourier expansion. An honourable first attempt is to let $h(\xi, \tau) = e^{-2\pi\iota\xi\tau}$. Then g becomes the Fourier transform of f, so the inducing points become *inducing frequencies*. If k is a stationary kernel with Fourier transform $\hat{k}$, then

$$\hat{f}^{(\mathrm{IDIP})}(t) = \sum_{i=1}^{M} \alpha_i \hat{k}(\xi_{u,i}) e^{-2\pi\iota\xi_{u,i}t}.$$

It appears that we have our desired result! Unfortunately not, because g turns out to be white noise with infinite variance for which inducing points do not work. To fix this, one could apply yet another inter-domain transform. By the Convolution Theorem, this is equivalent to applying a window to f before Fourier transforming. $L^2[a,b]$-VFFs choose a rectangular window on $[a,b]$, which learns f on only the finite interval $[a,b]$ instead of all of $\mathbb{R}$:

$$g(\xi) \mid f = \int_a^b f(t) e^{-2\pi\iota\xi(t-a)}\,\mathrm{d}t$$

where ξ should be harmonic on $[a,b]$. This works, sort of. The posterior predictive mean $\hat{f}^{(\mathrm{IDIP})}(t)$ is close to a Fourier sum for t far from the edges of $[a,b]$, but near the edges there is undesirable behaviour; this is illustrated in Figure 7.

Like $L^2[a,b]$-VFFs, VFFs also approximate the function on only a compact interval $[a,b]$, but do so without introducing edge effects. Their construction utilises the theory of reproducing kernel Hilbert spaces (RKHS). Since we consider a compact interval $[a,b]$, if we endow this interval with a finite measure μ, then Mercer's Theorem states that k can be written as

$$k(t,t') = \sum_{i=1}^{\infty} \lambda_i \phi_i(t) \phi_i^*(t')$$

where $(\lambda_i)_{i=1}^{\infty}$ are absolutely summable positive eigenvalues and $(\phi_i)_{i=1}^{\infty}$ orthonormal

eigenfunctions of the integral operator

$$T_k f = t \mapsto \langle k(t, \cdot), f \rangle, \qquad \langle f, g \rangle = \int_a^b f(t) g^*(t) \, \mathrm{d}\mu(t).$$

What does this get us? Well, if the inter-domain transform instead integrates against μ, then

$$\begin{aligned}
\hat{f}^{(\mathrm{IDIP})}(t) &= \sum_{i=1}^{M} \alpha_i \int_a^b k(t, \tau) h(\xi_{u,i}, \tau) \, \mathrm{d}\mu(\tau) \\
&= \sum_{i=1}^{M} \alpha_i \langle k(t, \cdot), h(\xi_{u,i}, \cdot) \rangle \\
&= \sum_{i=1}^{M} \alpha_i T_k(h(\xi_{u,i}, \cdot))(t).
\end{aligned}$$

That is, we wish to find an h such $T_k(h(\xi, \cdot))(t) = \psi(\xi, t) = e^{-2\pi \imath \xi t}$. And we know which functions pass through T_k unharmed: it's eigenfunctions! Therefore, suppose that $\psi(\xi, \cdot) \in \operatorname{span} \{\phi_i\}_{i=1}^{\infty}$. Then

$$\psi(\xi, \cdot) = \sum_{i=1}^{\infty} \phi_i(t) \langle \psi(\xi, \cdot), \phi_i \rangle,$$

so a good candidate is

$$h(\xi, t) = \sum_{i=1}^{\infty} \frac{1}{\lambda_i} \phi_i(t) \langle \psi(\xi, \cdot), \phi_i \rangle.$$

Indeed,

$$\begin{aligned}
T_k(h(\xi, \cdot))(t) &= \sum_{i=1}^{\infty} \frac{1}{\lambda_i} T_k(\phi_i)(t) \langle \psi(\xi, \cdot), \phi_i \rangle \\
&= \sum_{i=1}^{\infty} \phi_i(t) \langle \psi(\xi, \cdot), \phi_i \rangle \\
&= \psi(\xi, t) = e^{-2\pi \imath \xi t},
\end{aligned}$$

so we have found an inter-domain transform h that achieves the desired Fourier features, at last. Hensman et al. (2016) show that this construction works for the family of Matérn kernels of half-integer order (Rasmussen et al., 2006), and that all computations are

tractable. The resulting inter-domain inducing point method is called VFFs (or RKHS-VFFs). Figure 7 illustrates the predictive features obtain through this construction; note that, within $[a, b]$, the predictive features are exactly sinusoidal.

VFFs demonstrate good performance both in terms of computational speed and predictive power. See the original paper by Hensman et al. (2016) for details.

3.3 Rates of Convergence

Although VFFs demonstrate improved performance compared to RFFs, their performance gain comes at the cost of a much more complex construction. The simplicity of RFFs is one of its benefits, which, as we saw in Topic 2, makes it amenable to convergence analysis. For VFFs and inducing point methods in general, the question of how many inducing points are needed is a more complicated one to answer. Recently, Burt et al. (2018) showed that for a particular inter-domain method that uses *eigenfunction inducing features*, under certain conditions $M \propto \log N$ many inducing points suffice, thus achieving a significant asymptotic gain.

Use Mercer's Theorem to decompose

$$k(t, t') = \sum_{i=1}^{\infty} \lambda_i \phi_i(t) \phi_i^*(t').$$

Then the *eigenfunction inducing features* $(u_i)_{i=1}^{\infty}$ are given by

$$u_i \mid f = \frac{1}{\sqrt{\lambda_i}} \int f(t) \phi_i(t) \, \mathrm{d}\mu(t).$$

These eigenfunction inducing features behave very nicely, which is what makes them attractive to use:

$$\mathbb{E}[u_i u_j] = 1 \text{ if } i = j \text{ else } 0, \qquad \mathbb{E}[f(t_i) u_j] = \sqrt{\lambda_j} \phi_j(t_i).$$

The main result of Burt et al. (2018) is the following:

Theorem 3.1. Fix $\varepsilon > 0$ and $\delta > 0$. Let $(t_i)_{i=1}^{\infty}$ be sampled i.i.d. from $\mathcal{N}(0, \alpha)$, let k be an exponentiated-quadratic kernel, and let μ have density $\mathcal{N}(0, \beta)$ with $\beta > 2\alpha$. Then there are $\tilde{N}$ and $\tilde{C}$ such that, for all $N > \tilde{N}$, the inter-domain point method with

$M = \tilde{C} \log N$ eigenfunction inducing features achieves

$$\mathrm{D}_{\mathrm{KL}}(q(f) \,\|\, p(f \,|\, \mathcal{D})) \leq \varepsilon$$

with probability at least $1 - \delta$. ◀

Sketch of Proof. The key ingredient is the inequality

$$\mathrm{D}_{\mathrm{KL}}(q(f) \,\|\, p(f \,|\, \mathcal{D})) \leq \frac{c}{2\sigma^2}\left(1 + \frac{\|y\|^2}{\sigma^2 + c}\right),$$

where y are the observed values, σ^2 is the observation noise, and

$$c = \operatorname{tr}\left(K_{yy} - K_{yu} K_{uu}^{-1} K_{uy}\right),$$

where $(K_{ff})_{ij} = \mathbb{E}[f(t_i) f(t_j)]$, $(K_{fu})_{ij} = \mathbb{E}[f(t_i) u_j]$, and $(K_{uu})_{ij} = \mathbb{E}[u_i u_j]$. Therefore, assuming that $\|y\|^2 = O(N)$, we need to scale M with N such that $c = O(N^{-1})$. A calculation (try it!) shows that

$$\frac{1}{N} c = \sum_{m=M+1}^{\infty} \lambda_m \left[\frac{1}{N} \sum_{i=1}^{N} \phi_m^2(t_i)\right].$$

Using Chebychev's Inequality to upper bound the quantity between square brackets with high probability[4], we find that $M \propto \log N$ suffices since, for the EQ kernel, $\sum_{m=M+1}^{\infty} \lambda_m = O(A^M)$ for some $A \in (0, 1)$.

3.4 Conclusion

Whereas RFFs approximate the kernel and perform exact inference in this approximate model, the inducing point approximation for Gaussian processes attempts to bring an approximate posterior close to the true posterior. This eliminates the risk of overfitting otherwise present in the SSA. VFFs is a particular inter-domain inducing point method designed to combine the benefits of inducing points with the representative power of RFFs, where key in the design of the inter-domain transform are tools from RKHS theory. Moreover, those tools can be used to design inter-domain inducing point methods that are amenable to convergence analysis.

[4] For simplicity, we ignore the dependency of ϕ_m on m; see the original paper for details.

4 Topic 4: Spectrum Estimation

The power spectral density (PSD) is of interest in a wide range of disciplines, like natural sound processing, statistical physics, astrophysics, biomedical engineering, materials science, and telecommunications, among others. Typically only a limited number of noisy observations of the underlying signal are available, often sampled at irregularly spaced points in time; the main challenge in spectrum estimation (SE) is designing estimators of the PSD that deal with these issues.

Estimators can roughly be categorised as (1) parametric methods that assume a parametric form for the underlying signal, resulting in a parametric estimate of the PSD, or (2) nonparametric methods that do not assume any particular structure, like the squared modulus of the discrete Fourier transform (DFT). We have already seen examples of both categories: the Sparse Spectrum Approximation (SSA) assumes a line spectrum and the Spectral Mixture Kernel (SMK) assumes a Gaussian mixture, whereas the Gaussian Process Convolution Model (GPCM) assumes of nonparametric prior over the PSD through a cleverly transformed Gaussian process.

Tobar (2018) recently introduced a method for SE called Bayesian Nonparametric Spectral Estimation (BNSE). BNSE is a Gaussian-process-based model that properly accounts for uncertainty deriving from having only finitely many observations, handles noise and irregularly spaced observations, and is able to exploit structure in the underlying signal, all whilst retaining a closed-form inference. This model will be our final topic.

4.1 Bayesian Nonparametric Spectral Estimation

A first take on a Gaussian-process-based model for SE might be the following: Suppose that our underlying signal is generated by a GP: $f \sim \mathcal{GP}(0, k)$. Then the PSD is given by $s(\xi) \,|\, f = |\hat{f}(\xi)|^2$ where $\hat{f}$ denotes the Fourier transform of f. Thus, given noisy observations $e = (y(t_1), \ldots, y(t_n))$ of f where y is a noisy version of f, we could estimate the PSD with a conditional expectation: $\hat{s}(\xi) = \mathbb{E}[s(\xi) \,|\, e]$. Unfortunately, for k stationary, as often is the case, this does not work: samples from f are then not integrable almost surely, meaning that the Fourier transform of $\hat{f}$ does not converge.

Instead, we consider the spectrum of a *windowed* version of f: $f_w(t) \,|\, f = f(t)w(t)$

where w is the window. Then samples of f_w are almost surely integrable, so the Fourier transform $\hat{f}_w$ of $\hat{f}$ almost surely exists. The spectrum $\hat{f}_w$ of f_w is called a *local spectrum* of f. Use of window functions is common in SE; they can arise as a consequence of acquisition devices or may be used for algorithmic purposes, like here (Tobar, 2018).

The posterior over the PSD of the windowed version of f is the estimator proposed by Tobar (2018). The posterior mean gives a point estimate:

$$\hat{s}_w(\xi) = \mathbb{E}[s_w(\xi) \mid e].$$

To compute this expectation, note that

$$\begin{aligned}
\hat{s}_w(\xi) &= \mathbb{E}[|\hat{f}_w(\xi)|^2 \mid e] \\
&= \mathbb{E}[(\text{Re } \hat{f}_w(\xi))^2 \mid e] + \mathbb{E}[(\text{Im } \hat{f}_w(\xi))^2 \mid e] \\
&= \Big(\mathbb{V}[\text{Re } \hat{f}_w(\xi) \mid e] + \mathbb{E}^2[\text{Re } \hat{f}_w(\xi) \mid e]\Big) \\
&\qquad + \Big(\mathbb{V}[\text{Im } \hat{f}_w(\xi) \mid e] + \mathbb{E}^2[\text{Im } \hat{f}_w(\xi) \mid e]\Big),
\end{aligned}$$

where the posterior moments of the real and imaginary part of $\hat{f}_w$ follow from the usual conditioning formulas for Gaussian processes if the covariance functions of Re $\hat{f}_w$ and Im $\hat{f}_w$ and their cross-covariances with the observations can be computed. We let $w(t) = e^{-\alpha\pi^2 t^2}$ and choose k to be a Spectral Mixture Kernel (SMK), or in simple cases an exponentiated-quadratic (EQ) kernel. The choice of window facilitates tractability of the calculation of the posterior moments.

4.2 Posterior Moments of $\hat{f}_w$

The covariance functions of Re $\hat{f}_w$ and Im $\hat{f}_w$ and their cross-covariances with the observations can be derived from that of $\hat{f}_w$:

$$\begin{aligned}
k_{\text{Re } \hat{f}_w}(\xi, \xi') &= \frac{1}{2}(k_{\hat{f}_w}(\xi, \xi') + k_{\hat{f}_w}(\xi, -\xi')), \qquad & k_{y(\text{Re } \hat{f}_w)}(t, \xi) &= \text{Re } k_{y\hat{f}_w}(t, \xi), \\
k_{\text{Im } \hat{f}_w}(\xi, \xi') &= \frac{1}{2}(k_{\hat{f}_w}(\xi, \xi') - k_{\hat{f}_w}(\xi, -\xi')), \qquad & k_{y(\text{Im } \hat{f}_w)}(t, \xi) &= \text{Im } k_{y\hat{f}_w}(t, \xi),
\end{aligned}$$

where

$$k_{\hat{f}_w}(\xi, \xi') = \mathbb{E}[\hat{f}_w^*(\xi)\hat{f}_w(\xi')], \qquad k_{y\hat{f}_w}(t, \xi) = \mathbb{E}[y^*(t)\hat{f}_w(\xi)],$$

which we will compute now. Let $\mathcal{F}\{f\}(\xi)$ denote the Fourier transform of an integrable function f:

$$\mathcal{F}\{f\}(\xi) \equiv \int_{-\infty}^{\infty} f(t) e^{-2\pi\imath\xi^\mathsf{T} t} \,\mathrm{d}t,$$

where we may explicitly denote dependence on the variable t: $\mathcal{F}\{f\}(\xi) = \mathcal{F}_t\{f(t)\}(\xi)$. The main ingredient of the calculation is the Convolution Theorem:

$$\begin{aligned}
k_{\hat{f}_w}(\xi, \xi') &= \mathbb{E}[\hat{f}_w^*(\xi)\hat{f}_w(\xi')] \\
&= \mathbb{E}[\mathcal{F}_{t,t'}\{f(t)f(t')w(t)w(t')\}(-\xi, \xi')] \\
&= (\mathcal{F}_{t,t'}\{k(t-t')\}(u,u') * \mathcal{F}_{t,t'}\{w(t)w(t')\}(u,u'))(-\xi,\xi'), \\
&= (\hat{k}(u)\delta(u+u') * \hat{r}_w(u,u'))(-\xi,\xi'),
\end{aligned}$$

where u and u' denote the variables of integration in the convolution, and where $\hat{k}$ is the Fourier transform of k and $\hat{r}_w$ the Fourier transform of $(t,t') \mapsto w(t)w(t')$. The function $\hat{r}_w$ admits a nice closed form:

$$\hat{r}_w(\xi+u, \xi'+u) = \mathcal{N}(\xi - \xi'; 0, \alpha)\mathcal{N}\big(u; \tfrac{1}{2}(\xi+\xi'), \tfrac{1}{4}\alpha\big).$$

Then, writing the convolution as an integral,

$$\begin{aligned}
k_{\hat{f}_w}(\xi, \xi') &= \int \hat{k}(u)\delta(u+u')\hat{r}_w(-\xi-u, \xi'-u') \,\mathrm{d}u \,\mathrm{d}u' \\
&= \int \hat{k}(u)\hat{r}_w(\xi+u, \xi'+u) \,\mathrm{d}u \\
&= \mathcal{N}(\xi-\xi'; 0, \alpha)\Big(\hat{k}(u) * \mathcal{N}\big(u; 0, \tfrac{1}{4}\alpha\big)\Big)\big(\tfrac{1}{2}(\xi+\xi')\big).
\end{aligned}$$

The cross-covariance between $\hat{f}_w$ and y admits a similar calculation:

$$\begin{aligned}
k_{y\hat{f}_w}(t, \xi) &= \mathbb{E}[y^*(t)\mathcal{F}_{t'}\{f(t')w(t')\}(\xi)] \\
&= \mathcal{F}_{t'}\{k(t-t')w(t')\}(\xi) \\
&= (\hat{k}(u)e^{-2\pi\imath u t} * \mathcal{F}\{w\}(u))(\xi) \\
&= \Big(\hat{k}(u)e^{-2\pi\imath u t} * \mathcal{N}\big(u; 0, \tfrac{1}{2}\alpha\big)\Big)(\xi).
\end{aligned}$$

For kernels k other than the EQ kernel, convolution of $\hat{k}$ with a normal density might be intractable. In those cases, for small α, we may approximate these normal densities

by a Dirac delta, so the convolutions simplify:

$$k_{\hat{f}_w}(\xi, \xi') \approx \mathcal{N}(\xi - \xi'; 0, \alpha)\hat{k}\big(\tfrac{1}{2}(\xi + \xi')\big), \qquad k_{y\hat{f}_w}(t, \xi) \approx \hat{k}(\xi)e^{-2\pi\imath\xi t}.$$

4.3 Connection with the Discrete Fourier Transform

The posterior mean of the local spectrum is given by

$$\mathbb{E}[\hat{f}_w(\xi) \mid e] = \int \hat{k}(u)\left(\sum_{i=1}^{N} e^{-2\pi\imath u t_i}(K_e^{-1}e)_i\right)\mathcal{N}\big(u; \xi, \tfrac{1}{2}\alpha\big)\,\mathrm{d}u$$

where $(K_e)_{ij} = k_y(t_i, t_j)$. This can be interpreted as the discrete Fourier transform (DFT) of a whitened version of the observations, weighted by the PSD of f, which constitutes our prior knowledge, and finally smoothed due to windowing effects. If the prior is largely uninformative, $K_e \approx I$, then, as the window becomes infinitely wide, we recover a weighted DFT:

$$\lim_{\alpha \to 0} \mathbb{E}[\hat{f}_w(\xi) \mid e] \approx \hat{k}(\xi) \sum_{i=1}^{N} e^{-2\pi\imath\xi t_i} e_i$$

4.4 Comparison to the Lomb–Scargle Method

A related classical method for SE with irregularly sampled observations is the Lomb–Scargle (LS) method. To estimate the power at a frequency ξ, LS uses an empirical estimate of the power of the sum of a sine and cosine with frequency f fit to the data using least squares:

$$s(\xi) = \frac{A^2}{N}\sum_{i=1}^{n}\sin^2(2\pi\xi t_i - \phi) + \frac{B^2}{N}\sum_{i=1}^{n}\cos^2(2\pi\xi t_i - \phi),$$

$$(A, B) = \arg\min_{A', B'} \sum_{i=1}^{N}(y(t_i) - A'\sin(2\pi\xi t_i - \phi) - B'\cos(2\pi\xi t_i - \phi))^2$$

where the phase shift ϕ is chosen such that the sine and cosine evaluated at the data are mutually orthogonal. LS differs from BNSE in the following aspects: (1) BNSE assumes a probabilistic model for the data, which can encode prior knowledge and allows the spectrum to be stochastic. On the other hand, LS assumes a particular parametric model learned using least squares, which does not encode prior knowledge

and only generates a point estimate of the spectrum. (2) BNSE only has to do inference once, after which the spectrum can be estimated at any frequency, whereas LS needs to be retrained for new frequencies.

Probabilistically speaking, LS assumes that the underlying signal is a sum of sines and cosines fit with maximum likelihood, assuming i.d.d. Gaussian noise on the observed values. If one instead assumes a Gaussian prior over $A(\xi)$ and $B(\xi)$, then Tobar (2018) shows that in the limit a Gaussian process with a Spectral Mixture Kernel (SMK) can be recovered, thus recovering the generative model of BNSE.

4.5 Conclusion

Bayesian Nonparametric Spectral Estimation (BNSE) is a novel model for SE where, instead of a parametric model, a Gaussian process prior over the underlying signal is assumed. This way, due to the Bayesian nature of the model, uncertainty deriving from noise and having only finitely many observations is handled automatically. A unique advantage of BNSE is that the estimate of the PSD is a closed-form function, meaning that it can be efficiently optimised to find periodicities.

References

Blundell, Charles et al. (2015). "Weight uncertainty in neural networks". In: *arXiv preprint arXiv:1505.05424* (cit. on p. 15).

Brochu, E. et al. (Dec. 2010). "A Tutorial on Bayesian Optimization of Expensive Cost Functions, With Application to Active User Modeling and Hierarchical Reinforcement Learning". In: *arXiv preprint arXiv:1012.2599*. eprint: https://arxiv.org/abs/1012.2599 (cit. on p. 4).

Bruinsma, W. P. (2016). "The Generalised Gaussian Convolution Process Model". MA thesis. Department of Engineering, University of Cambridge. DOI: 10.17863/CAM.20389. URL: https://www.repository.cam.ac.uk/bitstream/handle/1810/273357/Bruinsma-MPhil-2016.pdf (cit. on p. 22).

Bui, T. D. et al. (May 2016a). "A Unifying Framework for Gaussian Process Pseudo-Point Approximations Using Power Expectation Propagation". In: *arXiv preprint arXiv:1605.07066*. eprint: https://arxiv.org/abs/1605.07066 (cit. on p. 28).

Bui, T. D. et al. (Feb. 2016b). "Deep Gaussian Processes for Regression Using Approximate Expectation Propagation". In: *arXiv preprint arXiv:1602.04133*. eprint: https://arxiv.org/abs/1602.04133 (cit. on p. 4).

Bui, Thang D. et al. (2014). "Tree-Structured Gaussian Process Approximations". In: *Advances in Neural Information Processing Systems* 27, pp. 2213–2221 (cit. on p. 28).

Burt, David et al. (2018). "Explicit Rates of Convergence for Sparse Variational Inference in Gaussian Process Regression". In: *Symposium on Advances in Approximate Bayesian Inference 1* (cit. on p. 33).

Chen, K. et al. (Aug. 2018). "Generalized Spectral Mixture Kernels for Multi-Task Gaussian Processes". In: *arXiv preprint arXiv:1808.01132*. eprint: https://arxiv.org/abs/1808.01132 (cit. on p. 20).

Cramer, Harald (1940). "On the Theory of Stationary Random Processes". In: *Annals of Mathematics* 41.1, pp. 215–230. URL: http://www.jstor.org/stable/1968827 (cit. on p. 10).

Cutajar, Kurt et al. (Aug. 2017). "Random Feature Expansions for Deep Gaussian Processes". In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup et al. Vol. 70. Proceedings of Machine Learning Research. PMLR, pp. 884–893 (cit. on pp. 14, 15).

Damianou, Andreas (2014). "Deep Gaussian Processes and Variational Propagation of Uncertainty". PhD thesis. Department of Neuroscience, University of Sheffield (cit. on p. 8).

Deisenroth, M. P. et al. (2011). "PILCO: A Model-Based and Data-Efficient Approach to Policy Search". In: ed. by L. Getoor et al. Vol. 28. Omnipress, pp. 465–472 (cit. on p. 4).

Duvenaud, D. et al. (Feb. 2014). "Avoiding Pathologies in Very Deep Networks". In: *arXiv preprint arXiv:1402.5836*. eprint: https://arxiv.org/abs/1402.5836 (cit. on p. 9).

Duvenaud, David (2014). "Automatic Model Construction With Gaussian Processes". PhD thesis. Computational and Biological Learning Laboratory, University of Cambridge (cit. on p. 4).

G. Matthews, A. G. d. et al. (Apr. 2015). "On Sparse Variational Methods and the Kullback-Leibler Divergence Between Stochastic Processes". In: *arXiv preprint arXiv:1504.07027*. eprint: https://arxiv.org/abs/1504.07027 (cit. on p. 28).

Gibbs, Mark N. (1997). "Bayesian Gaussian Processes for Regression and Classification". PhD thesis. Computational and Biological Learning Laboratory, University of Cambridge (cit. on p. 20).

Gredilla, Miguel Lázaro et al. (2009). "Inter-Domain Gaussian Processes for Sparse Inference Using Inducing Features". In: 22, pp. 1087–1095 (cit. on p. 29).

Hennig, Philipp et al. (2015). "Probabilistic Numerics and Uncertainty in Computations". In: *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences* 471.2179. ISSN: 1364-5021. DOI: 10.1098/rspa.2015.0142. URL: http://rspa.royalsocietypublishing.org/content/471/2179/20150142 (cit. on p. 4).

Hensman, J. et al. (Nov. 2016). "Variational Fourier Features for Gaussian Processes". In: *arXiv preprint arXiv:1611.06740*. eprint: https://arxiv.org/abs/1611.06740 (cit. on pp. 29, 30, 32, 33).

Kingma, D. P. et al. (Dec. 2013). "Auto-Encoding Variational Bayes". In: *arXiv preprint arXiv:1312.6114*. eprint: https://arxiv.org/abs/1312.6114 (cit. on p. 16).

Lázaro-Gredilla, Miguel et al. (2010). "Sparse Spectrum Gaussian Process Regression." In: *Journal of Machine Learning Research* 11, pp. 1865–1881. URL: http://dblp.uni-trier.de/db/journals/jmlr/jmlr11.html#Lazaro-GredillaCRF10 (cit. on p. 14).

LeCun, Yann et al. (2015). "Deep Learning". In: *Nature* 521.7553, pp. 436–444. DOI: 10.1038/nature14539. URL: https://doi.org/10.1038/nature14539 (cit. on p. 7).

MacKay, David J. C. (2002). *Information Theory, Inference & Learning Algorithms*. Cambridge University Press (cit. on p. 6).

Matthews, A. G. et al. (Apr. 2018). "Gaussian Process Behaviour in Wide Deep Neural Networks". In: *arXiv preprint arXiv:1804.11271*. eprint: https://arxiv.org/abs/1804.11271 (cit. on p. 8).

Minka, Tom (2000). *Deriving Quadrature Rules From Gaussian Processes*. Tech. rep. (cit. on p. 4).

Neal, Radford M. (1995). "Bayesian Learning for Neural Networks". PhD thesis (cit. on p. 8).

Parra, G. et al. (Sept. 2017). "Spectral Mixture Kernels for Multi-Output Gaussian Processes". In: *arXiv preprint arXiv:1709.01298*. eprint: https://arxiv.org/abs/1709.01298 (cit. on pp. 10, 18).

Rahimi, Ali et al. (2008). "Random Features for Large-Scale Kernel Machines". In: *Advances in Neural Information Processing Systems 20*. Ed. by J. C. Platt et al. Curran Associates, Inc., pp. 1177–1184. URL: http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf (cit. on p. 11).

Rasmussen, Carl Edward et al. (2006). *Gaussian Processes for Machine Learning*. MIT Press (cit. on pp. 4, 6, 7, 32).

Remes, S. et al. (May 2017). "Non-Stationary Spectral Kernels". In: *arXiv preprint arXiv:1705.08736*. eprint: https://arxiv.org/abs/1705.08736 (cit. on p. 21).

Titsias, Michalis K. (2009). "Variational Learning of Inducing Variables in Sparse Gaussian Processes". In: *Artificial Intelligence and Statistics* 12, pp. 567–574 (cit. on p. 27).

Tobar, Felipe (Sept. 2018). "Bayesian Nonparametric Spectral Estimation". In: *Advances in Neural Information Processing Systems 31*. Ed. by S. Bengio et al. Curran Associates, Inc. eprint: https://arxiv.org/abs/1809.02196 (cit. on pp. 35, 36, 39).

Tobar, Felipe et al. (2015). "Learning Stationary Time Series Using Gaussian Processes With Nonparametric Kernels". In: *Advances in Neural Information Processing Systems* 29, pp. 3501–3509 (cit. on p. 22).

Ulrich, Kyle et al. (2015). "GP Kernels for Cross-Spectrum Analysis". In: *Advances in Neural Information Processing Systems* 28, pp. 1999–2007 (cit. on p. 20).

Wilson, A. G. et al. (Feb. 2013). "Gaussian Process Kernels for Pattern Discovery and Extrapolation". In: *arXiv preprint arXiv:1302.4245*. eprint: `https://arxiv.org/abs/1302.4245` (cit. on pp. 17, 18).