Advances in Kernel Methods

Towards General-Purpose and Scalable Models

Yves-Laurent Kom Samo

Department of Engineering Science
University of Oxford

This dissertation is submitted for the degree of
Doctor of Philosophy

St. Anne's College June 2017

To my late father whom, 20 years on, I miss more than ever.

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. Notwithstanding the use of the first person of plural to express personal views throughout this thesis, this dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements.

Yves-Laurent Kom Samo
June 2017

Acknowledgements

First and foremost, I would like to thank the Oxford-Man Institute (OMI) for the relentless support I have received throughout the course of my DPhil, without which this thesis would likely not have been possible. In particular, I would like to extend profuse thanks to Terry Lyons who was the Director of the OMI at the time of my admission, and with whom I have had numerous intellectually stimulating and very enjoyable discussions at the intersection of pure mathematics, data science and philosophy. My stay at the OMI wouldn't have been the same without Marek Musiela, whose door is always open to any OMI students who could benefit from his immense wisdom of financial markets and his world-renowned expertise in stochastic analysis. Personally, I have been lucky to have many very fruitful discussions with Marek on stochastic analysis, finance and entrepreneurship, some of which have influenced my research and other personal endeavours. I would like to thank the great staff of the OMI, in particular Justin Sharp, Laura Okoli, and their teams, who have always warmly welcomed all my questions and requests. The IT facilities Justin's team manage are nothing but first class, and Laura's team have been of tremendous help in conference travel arrangements among other things.

I would like to thank my advisor Stephen Roberts who made my DPhil experience special, by giving me the freedom to drive my personal research agenda. I would also like to thank Google for sharing my enthusiasm for the research agenda I pursue in this thesis by awarding me a Google Fellowship in Machine Learning.

I am thankful for the love and camaraderie of my colleagues and friends. I will not attempt to name them all here for I would most likely forget important individuals. To all of you who have always genuinely been there for me I extend my sincere gratitude. I would like to extend special thanks to Jake Effoduh. His dynamism, kindness, indelible smile and indefectible commitment to be the voice of the unheard wherever he is have often swamped me in positivity, which undoubtedly has helped me in the course of my DPhil.

Finally, but most importantly, I would like to extend my heartfelt thanks to my family. I owe most of who I am today to my mother. Her strength after my father's

passing has been an inspiration to me in many ways. The abundant attention and care of my sister Patricia towards me have helped me through bumpy rides on numerous occasions. My thoughts also go to my younger cousins Hermann, Jordan and Emmanuel, whom I love more than brothers. Showing them that they can achieve any goals they put their minds to has been a motivation of mine for many years. Lastly, my thoughts go to my late father. I find great comfort in thinking that this thesis would have made him proud, which is perhaps my ultimate motivation in life.

Abstract

A wide range of statistical and machine learning problems involve learning one or multiple latent functions, or properties thereof, from datasets. Examples include regression, classification, principal component analysis, optimisation, learning intensity functions of point processes and reinforcement learning to name but a few. For all these problems, positive semi-definite kernels (or simply kernels) provide a powerful tool for postulating flexible nonparametric hypothesis spaces over functions. Despite recent work on such kernel methods, parametric alternatives, such as deep neural networks, have been at the core of most artificial intelligence breakthroughs in recent years. In this thesis, both theoretical and methodological foundations are presented for constructing *fully automated*, *scalable*, and *general-purpose* kernel machines that perform very well over a wide range of input dimensions and sample sizes. This thesis aims to contribute towards bridging the gap between kernel methods and deep learning and to propose methods that have the advantage over deep learning in performing well on both small and large scale problems.

In Part I we provide a gentle introduction to kernel methods, review recent work, identify remaining gaps and outline our contributions.

In Part II we develop *flexible* and *scalable* Bayesian kernel methods in order to address gaps in methods capable of dealing with the special case of datasets exhibiting locally homogeneous patterns. We begin with two motivating applications. First we consider inferring the intensity function of an inhomogeneous point process in Chapter 2. This application is used to illustrate that often, by carefully adding some mild asymmetry in the dependency structure in Bayesian kernel methods, one may considerably scale-up inference while improving flexibility and accuracy. In Chapter 3 we propose a scalable scheme for online forecasting of time series and *fully-online* learning of related model parameters, under a kernel-based generative model that is provably sufficiently flexible. This application illustrates that, for one-dimensional input spaces, restricting the degree of differentiability of the latent function of interest may considerably speed-up inference without resorting to approximations and without any adverse effect on flexibility or accuracy. Chapter 4 generalizes these approaches and

proposes a novel class of stochastic processes we refer to as *string Gaussian processes* (string GPs) that, when used as functional prior in a Bayesian nonparametric framework, allow for inference in linear time complexity and linear memory requirement, without resorting to approximations. More importantly, the corresponding inference scheme, which we derive in Chapter 5, also allows flexible learning of locally homogeneous patterns and automated learning of model complexity — that is automated learning of *whether* there are local patterns in the data in the first place, *how much* local patterns are present, and *where* they are located.

In Part III we provide a broader discussion covering all types of patterns (homogeneous, locally homogeneous or heterogeneous patterns) and both Bayesian or frequentist kernel methods. In Chapter 6 we begin by discussing what properties a family of kernels should possess to enable fully automated kernel methods that are applicable to any type of datasets. In this chapter, we discuss a novel mathematical formalism for the notion of 'general-purpose' families of kernels, and we argue that existing families of kernels are not general-purpose. In Chapter 7 we derive weak sufficient conditions for families of kernels to be general-purpose, and we exhibit tractable such families that enjoy a suitable parametrisation, that we refer to as *generalized spectral kernels* (GSKs). In Chapter 8 we provide a *scalable* inference scheme for *automated* kernel learning using *general-purpose* families of kernels. The proposed inference scheme *scales linearly* with the sample size and enables *automated learning of nonstationarity* and model complexity from the data, in virtually any kernel method. Finally, we conclude with a discussion in Chapter 9 where we show that deep learning can be regarded as a particular type of kernel learning method, and we discuss possible extensions in Chapter 10.

Table of contents

List of figures

List of tables

Nomenclature

Subscripts

CSS Complex-valued stationary Sparse Spectrum kernel

GS Generalized Spectral kernel

HGS Hybrid Generalized Spectral kernel

NSS Nonstationary Sparse Spectrum kernel

SGS Stationary Generalized Spectral kernel

SGS-MA Spectral Matérn kernel

SS Real-valued stationary Sparse Spectrum kernel

Acronyms / Abbreviations

pDGP p-Derivative Gaussian Process

pM-GP p-Markov Gaussian Process

ARD Automatic Relevance Determination

BCM Bayesian Commitee Machine

ESS Elliptical Slice Sampling

GAM Generalized Additive Models

GP Gaussian Process

gPoE Generalized Product of Experts

GPR Gaussian Process Regression

GSK Generalized Spectral Kernel

HGSK Hybrid Generalized Spectral Kernel

LGSSM Linear Gaussian State Space Model

MCMC Markov Chain Monte Carlo

MH Metrolis-Hastings

MoE Mixture of Experts

OGD Online Gradient Descent

PA Passive-Aggressive

rBCM Robust Bayesian Commitee Machine

RJ-MCMC Reversible-Jump Markov Chain Monte Carlo

RKHS Reproducing Kernel Hilbert Space

SGP String Gaussian Process

SVD Singular Value Decomposition

Part I

Introduction

Chapter 1

A Gentle Introduction to Kernel Methods

"There are no facts, only interpretations."

Friedrich Nietzsche

The presentation in this chapter purposely puts more emphasis on concepts than on mathematical technicalities for clarity. For a more detailed construction of the mathematical notions used herein, we refer the reader to Hofmann et al. (2008) and references therein.

Throughout the years kernel methods have found applications in nearly all types of machine learning tasks and applications. They indeed play a central role in regression problems (e.g. kernel ridge regression, Gaussian process regression), in classification problems (e.g. Support Vector Machines, Gaussian process classification), in clustering problems (e.g. kernel k-Means, spectral clustering), in factor models (e.g. kernel Principal Component Analysis, Gaussian Process Latent Variable Models), in optimization problems (e.g. Bayesian Optimization), in probabilistic integration (e.g. Bayesian Quadrature), in point process problems (e.g. Bayesian nonparametric learning of intensity functions of point processes) and time series analysis to name a few.

1.1 What Exactly are Kernel Methods?

Notwithstanding the specificities of each of the aforementioned techniques, it is safe to say that kernel methods are often introduced either to depart from linearity assumptions,

or to introduce a flexible notion of similarity between arbitrary objects such as images, graphs, semantic patterns and suchlike.

1.1.1 From Linear to Nonlinear Learning Machines

A major limitation of linear models is that they are inherently biased towards a specific data collection process, which might lead to poor performance. As an illustration, let us consider the standard linear regression model

$$y_i = \alpha^T x_i + \epsilon_i, \quad \alpha, x_i \in \mathbb{R}^d, \quad y_i, \epsilon_i \in \mathbb{R}, \quad \epsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2). \tag{1.1}$$

When an operator is tasked with explaining the phenomenon giving rise to responses y_i using the phenomenon underpinning covariates x_i, the above model would inevitably rely on *how* the covariates were obtained from the underlying phenomenon, rather than *the extent to which* the covariates characterise the underlying phenomenon. Indeed, had we collected the cubic, the exponential, or any other alternate function of these covariates in lieu of the covariates themselves, the resulting linear models would have provided completely different explanations of the phenomenon of interest. That being said, there is no reason *a priori* to believe that the collection of covariates was performed so as to provide a representation of the corresponding exogenous phenomenon that is linearly related to the response we wish to explain.

A solution to this pitfall is found by noting that the covariates x_i are simply a particular feature of the underlying phenomenon, and that additional features $\phi(x_i)$ might provide more insights into the problem. In other words, although the relationship between the response and collected covariates might not be linear, linearity might hold in a different feature space that could be learned from the data. The corresponding non-linear regression model, known as basis functions regression, may be written as

$$y_i = f(x_i) + \epsilon_i, \quad x_i \in \mathbb{R}^d, \quad y_i, \epsilon_i \in \mathbb{R}, \quad \epsilon_i \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, \sigma^2) \tag{1.2}$$

where

$$f(x) = \sum_{i=1}^{p} \alpha_i \phi_i(x), \quad p \in \mathbb{N} \tag{1.3}$$

and $\{\phi_1(x), \ldots, \phi_p(x)\}$ represents a possibly large number of features constructed using the collected covariates x.

Although the foregoing discussion was illustrated with linear regression, the argument applies more generally to any model aiming at learning a latent function of the

form $\alpha^T x$, and Equation (1.3) may provide a better alternative hypothesis space for the latent function of interest, especially when the feature maps (or basis functions) ϕ_i are learned from the data. Indeed, rather than postulating that linearity holds in the domain of the raw covariates, it might be preferable to *learn from the data* a domain in which linearity holds. It is worth noting that this setup is fairly general as we do not impose any restriction on the features ϕ_i at this point.

In some applications, the dimension of the feature space, p, might be required to be very large, possibly infinite, to be able to capture intricacies of the latent function. In such applications, it becomes impractical to learn the coefficients α_i directly, and yet one needs to be able to learn the latent function f from the data. What's more, as in any sound modelling framework, the method should abide by the principle of Occam's razor, which states that a model should not be more complex than necessary, even when the feature space is infinite-dimensional. The requirement imposed by Occam's razor has two consequences in our setup. First, we need a framework for evaluating how 'complex' a candidate function f in our hypothesis space is. Second, our approach should be nonparametric in the sense that the complexity of the best candidate function for our problem should be allowed to increase with the size of the training dataset. All these requirements are addressed by the mathematical notion of the *Reproducing Kernel Hilbert Space* (RHKS).

1.1.2 Reproducing Kernel Hilbert Spaces

Before defining RKHSs, we need to introduce the notion of positive semi-definite kernels (or simply kernels) and recall a standard decomposition result.

Definition 1.1 *Let $\mathcal{X}$ be a nonempty set. A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is said to be a positive semi-definite kernel when for all distinct $(x_1, \ldots, x_n) \in \mathcal{X}^n$, the matrix $[k(x_i, x_j)]_{1 \leq (i,j) \leq n}$ is symmetric positive semi-definite. k is said to be a (strictly) positive definite when additionally no matrix $[k(x_i, x_j)]_{1 \leq (i,j) \leq n}$ is singular.*

Theorem 1.2 *(Mercer's theorem) Let k be a continuous square integrable positive semi-definite kernel on a measure space $(\mathcal{X}, \mathcal{F}, \mu)$. There exists an orthonormal set $\{\phi_i\}_{i \in \mathbb{N}}$ of $L^2(\mathcal{X}, \mathcal{F}, \mu)$, and a sequence of non-negative values $\{\lambda_i\}_{i \in \mathbb{N}}$ such that*

$$k(x, y) = \sum_{i=1}^{+\infty} \lambda_i \phi_i(x) \phi_i(y). \tag{1.4}$$

Definition 1.3 *Let k be a continuous positive semi-definite kernel on a measure space $(\mathcal{X}, \mathcal{F}, \mu)$, and let $\{\phi_i, \lambda_i\}_{i \in I}$ be as per Theorem 1.2 where $I = \{i \in \mathbb{N} : \lambda_i > 0\}$.*

We denote the reproducing kernel Hilbert space with reproducing kernel k the set of functions

$$\mathcal{H}_k = \left\{ f : \mathcal{X} \to \mathbb{R} : f(x) = \sum_{i \in I} a_i \phi_i(x), \ \sum_{i \in I} \frac{a_i^2}{\lambda_i} < +\infty \right\}. \tag{1.5}$$

Moreover, $\mathcal{H}_k$ is a Hilbert space endowed with the inner product[1]

$$\left\langle \sum_{i \in I} a_i \phi_i, \sum_{j \in I} b_j \phi_j \right\rangle_{\mathcal{H}_k} := \sum_{i \in I} \frac{a_i b_i}{\lambda_i}. \tag{1.6}$$

In particular, it follows from Mercer's theorem that if we define $k_x : \mathcal{X} \to \mathbb{R}$ with $k_x(y) := k(x, y)$, then $k_x \in \mathcal{H}_k$ and

$$\forall f \in \mathcal{H}_k, \ \langle f, k_x \rangle_{\mathcal{H}_k} = f(x). \tag{1.7}$$

Going back to our original discussion, the functions $\{\sqrt{\lambda_i}\phi_i\}_{i \in I}$ play the role of the set of features, of which there could be infinitely many. Moreover, the squared norm $||f||^2_{\mathcal{H}_k} := \langle f, f \rangle_{\mathcal{H}_k}$ induced by the Hilbert space structure provides a suitable measure of complexity of candidate functions in the RKHS, and subsequently enables the development of methods that are consistent with Occam's razor. Furthermore, for a large class of machine learning tasks, aiming at learning a latent function, when the hypothesis space of candidate functions is a RKHS, the optimal solution (provably) takes the form

$$f^*(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i), \ \alpha_i \in \mathbb{R}, \tag{1.8}$$

where x_i are training covariates, regardless of the dimension of the feature space. Thus, the number of parameters we need to learn will grow with the size of the training sample rather than the dimension of the feature space.

1.1.3 RKHSs and Bayesian Nonparametrics

In previous sections we discussed frequentist machine learning methods for inferring a latent function, and we introduced RKHSs as suitable hypothesis spaces for functions. In most applications, it is be preferable to reason under uncertainty. To do so, we may regard the latent function as being random, or equivalently we may postulate *a priori* that it is a draw from a stochastic process whose law is referred to as *functional prior*. The learning procedure then consists of using evidence from training data to

[1] We use the symbol := to mean 'by definition'.

reduce our uncertainty about the latent function. This approach is known as *Bayesian nonparametrics*.

Although the hypothesis space of functions in the Bayesian nonparametrics approach, namely the space of paths of the functional prior, is not made explicit, it happens to have strong links with RKHSs. Indeed, a continuous function is the covariance function of a second-order stochastic process if and only if it is a positive semi-definite kernel. What's more, by the Karhunen-Loève expansion, a mean-zero second-order stochastic process f indexed on $\mathcal{X}$ and with continuous covariance function k may be decomposed as

$$f(x) = \sum_{i \in I} z_i \phi_i(x), \tag{1.9}$$

where I and ϕ_i are as per the previous section, and the random variables z_i are decorrelated. If $\mathcal{H}_k$, the RKHS with reproducing kernel k, is finite-dimensional, then the paths of f will lie in $\mathcal{H}_k$ with probability 1. When $\mathcal{H}_k$ is infinite-dimensional, the series

$$\sum_{i \in I} z_i^2 \tag{1.10}$$

will converge with probability 1, but the series

$$\sum_{i \in I} \frac{z_i^2}{\lambda_i} \tag{1.11}$$

might diverge, and as a result f might not lie in the RKHS. Nonetheless, our feature space narrative will still hold. Crucially, a mean-zero functional prior with continuous covariance function k encodes the same hypothesis set of features as the RKHS with reproducing kernel k.[2] The only possible departure between the RKHS approach and the Bayesian nonparametric approach is in the set of all allowed coordinates in the feature space.

1.1.4 Gaussian Processes

The most popular example of functional priors are mean-zero Gaussian processes. They arise when all finite-dimensional marginals are centred multivariate Gaussians. Gaussian processes are particularly convenient for analytical derivations as they are stable under common operations such as affine transformations, differentiation and

[2]To be precise, by 'encodes the same hypothesis set of features' we mean that functions in the RKHS with continuous reproducing kernel k and random draws from a mean-zero second order stochastic process with covariance function k can all be written as limit of linear combinations of the same set of functions.

integration. The Karhunen-Loève expansion of a mean-zero Gaussian process with continuous covariance function k reads

$$f(x) = \sum_{i \in I} z_i \phi_i(x), \quad z_i \sim \mathcal{N}(0, \lambda_i), \quad \forall i \neq j, z_i \perp z_j. \tag{1.12}$$

In the case of a Gaussian process, more can be said about its space of paths and the RKHS induced by its covariance function, notably the result due to Kallianpur (1970), that the paths of a Gaussian process either almost surely lie in the RKHS induced by its covariance function, or almost surely do not lie in the aforementioned RKHS.

1.1.5 Kernels for Similarity Measures

In nearly all machine learning tasks, the conceptual notion of similarity between inputs plays a central role. In supervised and semi-supervised learning, the basic assumption that underpins any learning method is that if two covariate vectors x_i and x_j are 'similar' then so should be their labels y_i and y_j. Unsupervised learning techniques such as the k-means and spectral clustering algorithms, are based on the premise that the more similar (or the closer) two covariate vectors x_i and x_j are, the more likely it is that they belong to the same class, again implying label similarity.

Given that labels are often real-valued or categorical, the notion of similarity between labels is unambiguous. However, similarity between covariates is less obvious. As an illustration, let us consider a regression problem where the exogenous variables are two-dimensional spatial locations. Clearly, we may parametrize the locations either using the Cartesian coordinates system or using the polar coordinates system. It might be tempting to use as similarity measure between covariate vectors x_i and x_j the inner product $x_i^T x_j$. Unfortunately, this would lead to a notion of similarity that is sensitive to the choice of coordinates system. Indeed, as illustrated in Figure 1.1, the set of coordinates that have a similarly of 1 with coordinates $(1, 1)$, namely

$$\{x \in \mathbb{R}^2 : x^T x^* = 1, \ x^* = (1, 1)\}, \tag{1.13}$$

corresponds to very different set of spatial locations depending on whether the covariates are defined via Cartesian or polar coordinates. Which coordinate system should be used, and subsequently which notion of similarity should be used, depends on the output response and should be learned from the data.

In our toy example, whether the covariates were gathered as polar or Cartesian coordinates, the desired notion of similarity may always be expressed as an inner

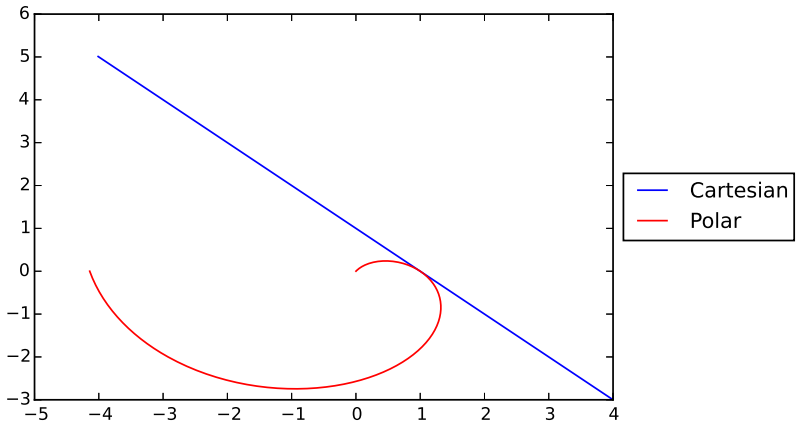

Fig. 1.1 Set of points that have an inner product of 1 with $(1, 1)$ in the polar domain (i.e. $\{r > 0, \theta \in [-\pi, \pi] : r + \theta = 1\}$) and in the Cartesian domain (i.e. $\{x, y \in \mathbb{R} : x + y = 1\}$), the former being projected back into the Cartesian domain using the standard map $x = r\cos\theta,\ y = r\sin\theta$.

product in an appropriate feature space. For instance, when it is preferable to work with the Cartesian notion of similarity, but covariates are polar coordinates, we may obtain the Cartesian similarity between (r_i, θ_i) and (r_j, θ_j) using the inner product on the feature space $\{\phi_1(r, \theta), \phi_2(r, \theta)\}$, namely

$$\phi_1(r_i, \theta_i)\phi_1(r_j, \theta_j) + \phi_2(r_i, \theta_i)\phi_2(r_j, \theta_j) \tag{1.14}$$

where $\phi_1(r, \theta) = r\cos\theta$ and $\phi_2(r, \theta) = r\sin\theta$. When the polar similarity is required but covariates are Cartesian coordinates, we may obtain the polar similarity between (x_i, y_i) and (x_j, y_j) using an inner product on a different feature space $\{\phi_1(x, y), \phi_2(x, y)\}$, namely

$$\phi_1(x_i, y_i)\phi_1(x_j, y_j) + \phi_2(x_i, y_i)\phi_2(x_j, y_j) \tag{1.15}$$

where $\phi_1(x, y) = \sqrt{x^2 + y^2}$ and $\phi_2(x, y) = 2\arctan\frac{y}{x+\sqrt{x^2+y^2}}$.

More generally, so long as an appropriate feature space is constructed or learned for the problem of interest, we may depart from the representation provided by the raw covariates, and define a more suitable notion of similarity based on the inner product in the feature space. Denoting $\mathbf{\Phi} : \mathcal{X} \to \mathbb{R}^p,\ \mathbf{\Phi}(x) = (\phi_1(x), \ldots, \phi_p(x))$ a continuous map from the original covariates space to the feature space, it is easy to verify from first principles that $k(x, y) := \mathbf{\Phi}(x)^T\mathbf{\Phi}(y)$ defines a continuous positive semi-definite kernel. Reciprocally, if k denotes a continuous positive semi-definite kernel, then by

Mercer's theorem it can be written as $k(x, y) := \mathbf{\Phi}(x)^T \mathbf{\Phi}(y)$ where the feature map is $\mathbf{\Phi}(x) = (\dots, \sqrt{\lambda_i}\phi_i(x), \dots)$ with $\{\lambda_i, \phi_i\}_{i \in \mathbb{N}}$ being as per Mercer's theorem.[3]

In summary, learning a similarity measure and learning a feature space can both be regarded as kernel learning.

1.1.6 Elements of Kernel Methods

So far we have assumed that an appropriate continuous kernel was given. However, it is crucial that the kernel be learned from the data as it encodes the set of features that are relevant for the problem of interest.

In practice, the kernel will be considered from a parametric family of positive semi-definite kernels $\{k_\theta, \theta \in \Theta\}$, which we will refer to as the *kernel generator*. Depending on the application, the kernel generator can be regarded as a set of candidate hypothesis spaces over functions, a set of candidate feature spaces, or a set of candidate similarity measures. In addition to the kernel generator, kernel methods also require a *loss function*, a *function learning algorithm*, and a *kernel learning algorithm*.

The *loss function*, which can also be thought of as an unormalized negative log-likelihood, provides a framework for evaluating a given candidate solution to the problem of interest. In regression problems, examples include the squared loss

$$l_{\text{squared}}(f(x), y) = (y - f(x))^2, \tag{1.16}$$

and the ϵ-insensitive loss

$$l_\epsilon(f(x), y) = \max\left(|y - f(x)| - \epsilon, 0\right). \tag{1.17}$$

Another example for binary classification problems with classes $y \in \{-1, 1\}$ is the hinge loss

$$l_{\text{hinge}}(f(x), y) = \max\left(1 - yf(x), 0\right). \tag{1.18}$$

When the objective of the problem is to infer a latent function, the *function learning algorithm* provides a framework for leveraging the information provided by training data to select an 'optimal' candidate function in the hypothesis space of functions encoded by a given kernel k_θ in the kernel generator. The optimality criteria used typically provides a trade-off between maximising the *model fit* and minimising the

[3] In a slight abuse of notation, we use the transpose operator to denote both the innner product on $\mathbb{R}^p$ when $p < +\infty$, and the inner product between squared integrable sequences when the feature space is infinite-dimensional.

model complexity, the framework for assessing the model fit being provided by the loss function. Two commonly used function learning algorithms are *regularized Empirical Risk Minimization* (ERM) and *posterior mean inference*.

Regularized ERM solves the following optimization problem:

$$f^* = \underset{f \in \mathcal{H}_{k_\theta}}{\operatorname{argmin}} \underbrace{\frac{1}{n} \sum_{i=1}^{n} l\left(f(x_i), y_i\right)}_{\text{Model fit}} + \underbrace{\lambda ||f||^2_{\mathcal{H}_{k_\theta}}}_{\text{Model complexity}}, \ \lambda \geq 0 \tag{1.19}$$

where $\mathcal{H}_{k_\theta}$ is the RKHS with reproducing kernel k_θ. By a result known as the *representer theorem* (Schölkopf and Smola (2001)), the solution to this problem always takes the form

$$f^*(x) = \sum_{i=1}^{n} \alpha_i k_\theta(x, x_i), \tag{1.20}$$

and consequently $||f^*||^2_{\mathcal{H}_{k_\theta}} = \boldsymbol{\alpha}^T \mathbf{K}_\theta \boldsymbol{\alpha}$ with $\mathbf{K}_\theta = [k_\theta(x_i, x_j)]_{1 \leq i,j \leq n}$ and $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_n)$. Thus, learning the optimal candidate f^* is equivalent to learning $\boldsymbol{\alpha}$, which may be done either analytically in some special cases, or numerically, typically using gradient-based methods.

Posterior mean inference consists of evaluating the optimal candidate function at some test inputs, namely $\mathbf{f}^*$, as the expectation of the posterior distribution

$$p(\mathbf{f}^*|\mathcal{D}, k_\theta) \propto \int \underbrace{p(\mathcal{D}|\mathbf{f}, k_\theta)}_{\text{Model fit}} \underbrace{p(\mathbf{f}^*, \mathbf{f}|k_\theta)}_{\text{Model complexity}} d\mathbf{f} \tag{1.21}$$

where $\mathcal{D}$ represents the training dataset, a mean-zero functional prior with covariance function k_θ is placed on the latent function f and $p(\mathbf{f}^*, \mathbf{f}|k_\theta)$ denotes its marginal at test and training inputs. Depending on the problem, $E\left(\mathbf{f}^*|\mathcal{D}, k_\theta\right)$ might be available analytically (e.g. Gaussian process regression — see for instance Rasmussen and Williams (2005)), obtained using sampling techniques (Gelman et al. (2013)) or approximated using variational methods (Bishop (2007b)).

Finally, as for the *kernel learning algorithm*, it aims at selecting an optimal kernel k_{θ^*} in the kernel generator $\{k_\theta, \theta \in \Theta\}$. This step is akin to *model selection*. Example approaches include selecting the kernel that accounts best for the training data, for instance by maximising the model evidence (sometimes called the marginal likelihood)

$$\theta^* = \underset{\theta \in \Theta}{\operatorname{argmax}}\ p(\mathcal{D}|k_\theta), \tag{1.22}$$

selecting the kernel whose solution to the problem of interest is the best, for instance by minimising the regularized empirical risk

$$\theta^* = \underset{\theta\in\Theta}{\operatorname{argmin}} \min_{f\in\mathcal{H}_{k_\theta}} \frac{1}{n}\sum_{i=1}^{n} l\left(f(x_i), y_i\right) \;+\; \lambda||f||^2_{\mathcal{H}_{k_\theta}},\ \lambda \geq 0, \tag{1.23}$$

or, when the kernel generator is large enough such that overfitting may occur, selecting a kernel that provides a suitable trade-off between performance and simplicity, for instance

$$\theta^* = \underset{\theta\in\Theta}{\operatorname{argmax}}\ p(\mathcal{D}|k_\theta)p(\theta) \tag{1.24}$$

where $p(\theta)$ is a prior, or

$$\theta^* = \underset{\theta\in\Theta}{\operatorname{argmin}} \left(\min_{f\in\mathcal{H}_{k_\theta}} \frac{1}{n}\sum_{i=1}^{n} l\left(f(x_i), y_i\right) \;+\; \lambda||f||^2_{\mathcal{H}_{k_\theta}}\right) + \gamma P(\theta),\ \lambda, \gamma \geq 0 \tag{1.25}$$

where $P(\theta)$ is a penalty term. The regularisation in Equation (1.25) also mitigates the issue that the RKHS norms in Equation (1.23), might not be comparable.

1.2 Limitations of Kernel Methods

Despite the wide applicability of kernel methods, pitfalls remain that prevent their widespread adoption in modern, large scale applications. We consider these drawbacks below.

1.2.1 Choice of Family of Kernels

Many families of kernels have been proposed that may be used as kernel generators. Considering that the kernel generator is the set of hypothesis feature spaces, it is of utmost importance that it be 'flexible' enough to ensure that appropriate feature spaces may be *learned* from the data. This poses two fundamental challenges: what mathematical formalism can we use to decide if a family of kernels is 'flexible enough' and how do we go about constructing such families of kernels? These two questions remain largely unanswered and unfortunately the choice of kernel families is often left to the user.

Attempts have been made to provide a mathematical framework to describe if the RKHS induced by a given kernel is sufficiently flexible. Unfortunately the resulting notions, namely the *universal* and $(L, p)-$*rich* properties (Steinwart and Christmann

(2008); Steinwart et al. (2006)), are not discriminative enough. Indeed, universality implies $(L, p)-$richness for continuous loss functions (Steinwart et al., 2006, Corollary 1), and for nearly every family of kernels commonly used in practice, each element in the family admits a spectral density and consequently is universal (Micchelli et al., 2006, Proposition 16). However, it is well known that the choice of family of kernels, and more importantly the choice of kernel in the family (i.e. the choice of kernel hyper-parameters), vastly affect the performance of kernel methods.

Families of kernels have been proposed that increase flexibility by combining simpler kernels, for instance popular vanilla kernels or kernels operating on a subset of the covariates, through operations that preserve positive semi-definitiveness. Examples include *multiple kernel learning* (Gönen and Alpaydın (2011)), *hierarchical kernel learning* (Bach (2008)), *additive kernels* (Duvenaud et al. (2011)), *compositional kernel search* (Duvenaud et al. (2013)) and *spectral mixture kernels* (Wilson and Adams (2013)). However, these methods are limited in that they predominantly give rise to stationary (i.e. translation-invariant) kernels, and there is no theoretical guarantee that the corresponding kernel generators are sufficiently flexible.

1.2.2 Nonstationarity Learning

Given that most stationary kernels (e.g. the Gaussian kernel, Matérn kernels, the rational quadratic kernel etc) are *universal* (i.e. their RKHSs contain functions that can approximate any continuous function on a compact input space to an arbitrarily small precision), their RKHSs, when used as a hypothesis space of functions, are flexible enough. However, this is not to say that we do not need nonstationary kernels.

Indeed, the universality of these kernels means that, *so long as one uses 'the right' function learning algorithm*, any continuous latent function can be learned from the data when the input space is compact (as it is often the case in practice). Having said that, 'the right' function learning algorithm in this context might depend on the data, and there is no reason *a priori* to believe that it can even be formulated as a solution to a statistics or optimization problem. When the function learning algorithm is given, the combination (*stationary universal kernel, function learning algorithm*) can be restrictive. As an illustration, when the kernel is stationary and the *representer theorem* (Schölkopf and Smola (2001)) applies, for instance in the case of regularized ERM or Gaussian process regression, the optimal solution takes the form

$$f^*(x) = \sum_{i=1}^{n} \alpha_i h(x - x_i), \tag{1.26}$$

with $k(x, x_i) := h(x - x_i)$ and where x_i are the training covariates. In other words the solution is always a linear combination of translations of the same baseline function h. Clearly, for the optimal latent function f^* to be as flexible as guaranteed by the universality property, the number of training samples n has to be sufficiently large. This is especially true if the latent function we wish to learn exhibits local patterns. However, in practice the operator rarely has control over the amount of data available, so that the combination (*stationary universal kernel, function learning algorithm*) might not be sufficiently flexible.

So, how can we go about increasing flexibility? We can either develop new function learning algorithms or relax the stationarity assumption. If we choose the former approach, the function learning algorithm cannot have solutions of the form $f^*(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$, otherwise the foregoing argument would still apply. Yet, it has to flexibly navigate a possibly infinite-dimensional RKHS to uncover an optimal function based on evidence from the data. Moreover, it would not be practical to learn the coordinates of the latent function in the feature space provided by Mercer's theorem as there could be infinitely many coordinates, and the feature map might not even be available analytically in the first place. Increasing flexibility by developing new function learning algorithms seems like a daunting task, and relaxing the stationarity assumption should be preferred.

That being said, existing nonstationary approaches such as the input-dependent rescaling of stationary kernels of Paciorek and Schervish (2004) and spatial deformation of stationary kernels (Damian et al. (2001); Sampson and Guttorp (1992); Schmidt and O'Hagan (2003)) are unfortunately application-specific. In order for a family of kernels to be guaranteed to work well with virtually every kernel method on any dataset, it should provably be able to encode any possible feature space of practical interest; in particular it should contain both stationary and nonstationary kernels, and whether nonstationarity is warranted on a given task should be *learned* from the data.

1.2.3 Scalability

Another limitation of kernel methods is that they scale poorly with the number of training samples. When the RKHSs of kernels in the kernel generator are infinite-dimensional, as it is often the case for popular kernels, commonly used function learning algorithms, such as regularized ERM and posterior mean inference under a Gaussian process prior, result in a memory requirement that grows quadratically with the number of training data, and a time complexity that often grows cubically with the number

of training data. These time complexity and memory requirements make the naive implementation of kernel methods impractical for large scale problems.

The root cause for this scalability issue is the lack of structure in the problem formulation and/or its solution. In regularized ERM for instance, or more generally when the *representer theorem* (Schölkopf and Smola (2001)) applies, the optimal solution is of the form

$$f^*(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i). \tag{1.27}$$

Hence, when one additional training point x_{n+1} is added, we need to compute all the $n+1$ additional terms

$$\langle k_{x_{n+1}}, k_{x_j} \rangle_{\mathcal{H}_k} = k(x_j, x_{n+1}), \ j \leq n+1, \tag{1.28}$$

in order to determine its impact on the RKHS norm of the optimal function, and subsequently its impact on the regularized empirical risk. As a result, the marginal increases in time complexity and memory requirement are at least linear in n, even though intuitively one would expect the marginal contribution of the $(n+1)$-th training point to overall performance to decrease with n asymptotically. Posterior mean inference under a mean-zero Gaussian process prior suffers from a similar issue. In effect, there is no structure in the functional prior other than the fact that marginals $p(f(x_1), \ldots, f(x_n))$ are mean-zero multivariate Gaussians with a *full* covariance matrix $[k(x_i, x_j)]_{1 \leq i,j \leq n}$. Once again, one would expect that an additional training point x_{n+1} would have a marginal impact on overall performance that decreases asymptotically, and yet the increase in memory (resp. time complexity) required to compute the new Gaussian probability density function grows linearly (resp. quadratically) with n.

Solutions have been proposed for scaling-up inference via structured approximations of the kernel. Random Fourier features methods (Le et al. (2013); Rahimi and Recht (2007); Yang et al. (2015)), for instance, consist of approximating the implicit and possibly infinite-dimensional feature space of a kernel k with an explicit finite-dimensional feature space so that we may write

$$k(x, y) \approx \mathbf{\Phi}(x)^T \mathbf{\Phi}(y), \quad \mathbf{\Phi}(x) \in \mathbb{R}^p, \quad p \ll n. \tag{1.29}$$

In a regularized ERM, the optimal solution then takes the form

$$f^*(x) = \sum_{i=1}^{n} \alpha_i \mathbf{\Phi}(x_i)^T \mathbf{\Phi}(x), \tag{1.30}$$

with squared RKHS norm given by

$$||f^*||^2_{\mathcal{H}_k} = \sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \mathbf{\Phi}(x_i)^T \mathbf{\Phi}(x_j)\alpha_j = \left(\sum_{i=1}^{n} \alpha_i \mathbf{\Phi}(x_i)\right)^T \left(\sum_{i=1}^{n} \alpha_i \mathbf{\Phi}(x_i)\right). \tag{1.31}$$

Whence the marginal increase in memory requirement and time complexity, due to adding a training point x_{n+1}, is proportional to the number of features p, but no longer depends on n. More generally, random Fourier features methods have the advantage of reducing the overall memory requirement of kernel methods from $\mathcal{O}(n^2)$ to $\mathcal{O}(np)$, and reducing the corresponding time complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(np^2)$, and they often come with theoretical guarantees. Unfortunately, existing random Fourier features approximations are only applicable to stationary kernels.

Approaches have also been proposed to improve the scalability of Bayesian kernel methods, typically involving Gaussian process priors. The two predominant ideas consist of introducing an approximating conditional independence structure in the finite dimensional Gaussian prior $p(f(x_1), \ldots, f(x_n))$, or allocating small subsets of the dataset to multiple learning agents whose predictions are then blended afterwards.

The typical approach for implementing the former idea consists of introducing so called 'inducing points' $(\hat{x}_1, \ldots, \hat{x}_p)$ and making an approximation of the form

$$\begin{aligned} &p(f(x_1), \ldots, f(x_n), f(\hat{x}_1), \ldots, f(\hat{x}_p)) \\ &\approx p\left(f(\hat{x}_1), \ldots, f(\hat{x}_p)\right) \hat{p}\left(f(x_1), \ldots, f(x_n) | f(\hat{x}_1), \ldots, f(\hat{x}_p)\right), \; p \ll n, \end{aligned} \tag{1.32}$$

where $f(\hat{x}_1), \ldots, f(\hat{x}_p)$ are fully dependent and consistent with the underlying functional prior, but $f(x_1), \ldots, f(x_n)$ enjoy some conditional independence structure given $f(\hat{x}_1), \ldots, f(\hat{x}_p)$. This alleviates the need to store and invert an $n \times n$ covariance matrix, and may yield memory requirement and time complexity that grow in $\mathcal{O}(np)$ and $\mathcal{O}(np^2)$ respectively. Unfortunately, this approximation, often referred to as a sparse Gaussian process (Quinonero-Candela and Rasmussen (2005)), does not deal with the choice of the number of inducing points p, though this crucially affects both computational gains and approximation accuracy. Moreover, the quality and the convergence of these approximations are seldom analysed in the literature.

The second concept underpins methods such as the piecewise Gaussian process of Kim et al. (2005), the treed Gaussian process of Gramacy and Lee (2008), the Bayesian Committee Machine of Tresp (2000) and the distributed Gaussian process of Deisenroth and Ng (2015). The computational gain here stems from the fact that storing and inverting multiple small covariance matrices, based on subsets of the

dataset, is computationally cheaper than storing and inverting a full covariance matrix based on the whole dataset. However, these approximations come at the cost of reduced accuracy, excessive posterior uncertainty, and they may lead to discontinuous and/or inconsistent predictions.

Occasionally, it may also be possible to exploit the geometry of the set of training inputs to scale-up inference. For instance, when the kernel is a separable function and the set of training inputs form a grid (Cartesian product), Gram matrices may be written as Kronecker products, thereby leading to more efficient storage, faster matrix vector multiplications, and faster computation of determinants and inverses (See Saatchi, 2011, Chap. 5). Moreover, if the covariance matrix is stationary, the input space is one-dimensional, and inputs are equally spaced, then the Gram matrix is Toeplitz, and operations such as matrix inversion and singular value decomposition may be performed with time complexity $\mathcal{O}(n^2)$ compared to $\mathcal{O}(n^3)$ in the general case.[4] Methods have also been proposed to interpolate the Gram matrix on a uniform or Cartesian product grid in order to benefit from some of the computational gains of Toeplitz and Kronecker matrices even when the input space is not structured (Wilson and Nickisch (2015)). However, none of these solutions is general as they require that either the kernel be separable (Kronecker techniques), or the kernel be stationary and the input space be one-dimensional (Toeplitz techniques).

1.3 Outline and Contributions

The overall aim of this thesis is to extend the theoretical and methodological foundations required for a widespread adoption of kernel methods. The rest of the thesis is grouped into two parts.

In Part II, flexible and scalable Bayesian kernel methods are developed for learning functions with locally homogeneous patterns from datasets. We begin with two motivating applications.

First in Chapter 2 we consider inferring the intensity function of an inhomogeneous point process, which is often thought to be a *doubly-intractable* problem. We use this application to illustrate that often the lack of scalability is the result of excessive regularity and/or excessive symmetry assumptions, and that by carefully adding some mild asymmetry to the dependency structure in Bayesian kernel methods, one may considerably scale-up inference while simultaneously improving accuracy.

[4] In the special case where a Toeplitz covariance matrix is circulant, the time complexity can be further reduced to $\mathcal{O}(n \log n)$.

In Chapter 3 we consider dynamic systems whose characteristics might evolve with time. We propose a scalable scheme for online forecasting of time series and *fully-online* learning of related model parameters under a Bayesian kernel-based generative model that is provably sufficiently flexible. In doing so, we aim to illustrate that, for one-dimensional input spaces, restricting the degree of differentiability of the latent function of interest may considerably speed-up inference without resorting to approximations and without any adverse effect on flexibility or accuracy.

Chapter 4 builds on these approaches to propose a novel class of stochastic processes we refer to as *string Gaussian processes* (string GPs) that, when used as functional prior in a Bayesian nonparametric framework, allow for exact inference in linear time complexity and with linear memory requirement. Unlike sparse Gaussian process methods, the scalability of string GP models is rooted in their theoretical construction, rather than resorting to *ex-post* approximations. More importantly, the corresponding inference scheme, which is derived in Chapter 5, also allows flexible learning of locally homogeneous patterns and automated learning of model complexity.

Part III provides a broader discussion of inferring latent functions or similarity measures from datasets in the presence of any type of patterns (homogeneous, locally homogeneous or heterogeneous) and using any type of kernel method (Bayesian or frequentist). In Chapter 6 we begin by proposing a novel mathematical formalism for the notion of 'general-purpose' families of kernels that provides performance guarantees, and we argue that existing families of kernels are not general-purpose. In Chapter 7 we derive weak sufficient conditions for families of kernels to be general-purpose, and construct tractable families that enjoy a simple parameterization, that we refer to as *generalized spectral kernels* (GSKs). In Chapter 8 we provide a *scalable*, *fully-automated* and *general-purpose* inference scheme for virtually any kernel method. In Chapter 9 we discuss the differences between deep learning and kernel methods, and we provide a sense in which deep learning may be regarded as a special type of kernel method. Finally, we conclude with a discussion in Chapter 10.

We choose to distribute detailed literature reviews across subsequent Chapters, so as to locally provide a context for each contribution we make.

1.4 Prerequisites

Although we aim for this thesis to be as self-contained as possible, at times, we will assume that the reader's commands of measure theory, probability theory, stochastic processes, Bayesian statistics, Monte Carlo Markov Chain (MCMC) and point processes,

are those of graduate students in applied mathematics and statistics. We refer the reader to Halmos (1950) for a detailed review of measure theory, and to Williams (1991) and Loève (1963) for an introduction to probability theory. For Gaussian processes notions, we refer the reader to MacKay (1998) and Rasmussen and Williams (2005) for a machine learning perspective, and to Adler and Taylor (2011) for a more theoretical exposition. As for Bayesian statistics and MCMC notions, we refer the reader to Chapter 5 of Murphy (2012) for an introduction to Bayesian statistics, and to Gelman et al. (2013) and Brooks et al. (2011) for a review of MCMC techniques. Finally, for point process notions, we refer the reader to Kingman (1993) for a gentle introduction, and to Daley and Vere-Jones (2008) for a more detailed theoretical exposition.

Part II

Flexible and Scalable Learning of Locally Homogeneous Patterns

Chapter 2

Scalable Nonparametric Bayesian Inference on Point Processes

> "Plurality is not to be posited without necessity."
>
> *William of Ockham*

The main objective of this chapter is to illustrate that the lack of scalability in Bayesian kernel methods can be due to postulating excessive regularity and/or symmetry assumptions in the construction of the prior. To do so, we propose a novel method for making *scalable* and *accurate* Bayesian nonparametric inference on the intensity function of a Poisson process, despite this problem being widely regarded as *doubly-intractable*.[1]

2.1 Introduction

Point processes are a standard model when the objects of study are the number and repartition of otherwise identical points on a domain, usually time or space. The Poisson point process is probably the most commonly used point process. It is fully characterised by an intensity function that is inferred from the data. Gaussian processes have been successfully used to form a prior over the (log-) intensity function for applications such as astronomy (Gregory and Loredo (1992)), forestry (Heikkinen and Arjas (1999)), finance (Basu and Dassios (2002)), and neuroscience (Cunningham et al. (2007)).

[1] The contributions in this chapter have been published in the proceedings of The 32nd International Conference on Machine Learning (ICML 2015).

In this chapter, we offer extensions to existing work as follows: we develop an exact nonparametric Bayesian model that enables efficient inference on Poisson processes. Our method scales linearly with the number of data points and does not resort to gridding the domain. We derive a MCMC sampler for core components of the model and show that our approach offers a faster and more accurate solution, as well as producing less correlated samples, compared to other approaches on both real-life and synthetic data.

2.2 Related Work

Nonparametric inference on point processes has been extensively studied in the literature. Rathbum and Cressie (1994) and Moeller et al. (1998) used a finite-dimensional piecewise constant log-Gaussian for the intensity function. Such approximations are limited in that the choice of the grid on which to represent the intensity function is arbitrary and one has to trade-off precision with computational complexity and numerical accuracy, with the complexity being cubic in the precision and exponential in the dimension of the input space. Both Kottas (2006) and Kottas and Sansó (2007) used a Dirichlet process mixture of Beta distributions as prior for the normalised intensity function of a Poisson process. Cunningham et al. (2008) proposed a model using Gaussian processes evaluated on a fixed grid for the estimation of intensity functions of renewal processes with log-concave renewal distributions. They turned hyper-parameters inference into an iterative series of convex optimization problems, where ordinarily cubic complexity operations such as Cholesky decompositions are evaluated in $\mathcal{O}(n \log n)$ leveraging the uniformity of the grid and the log-concavity of the renewal distribution. Adams et al. (2009) proposed an exact Markov Chain Monte Carlo (MCMC) inference scheme for the posterior intensity function of a Poisson process with a sigmoid Gaussian prior intensity, or equivalently a Cox process (Cox (1955)) with sigmoid Gaussian stochastic intensity. The authors simplified the likelihood of a Cox process by introducing latent so-called *thinning points*. The proposed scheme has a complexity exponential in the dimension of the input space, cubic in the number of data and thinning points, and performs particularly poorly when the data are sparse. Rao and Teh (2011) used *uniformization* to produce exact samples from a nonstationary renewal process whose hazard function is modulated by a Gaussian process, and consequently proposed an MCMC sampler to sample from the posterior intensity of a unidimensional point process. Although the authors have illustrated that their model is faster than that of Adams et al. (2009) on some synthetic and real-life

data, their method still scales cubically in the number of thinning and data points, and is not applicable to data in dimension higher than 1, such as spatial point processes.

2.3 Model Construction

2.3.1 Setup

We are tasked with making nonparametric Bayesian inference on the intensity function of a Poisson point process assumed to have generated a dataset $\mathcal{D} = \{s_1, ..., s_n\}$. To simplify the discourse without loss of generality, we will assume that data points take values in $\mathbb{R}^d$. W recall that a *Poisson point process* (PPP) on a bounded domain $\mathcal{S} \subset \mathbb{R}^d$ with non-negative *intensity function* λ is a locally finite random collection of points in $\mathcal{S}$ such that the numbers of points occurring in disjoint parts B_i of $\mathcal{S}$ are independent and each follows a Poisson distribution with mean $\int_{B_i} \lambda(s)ds$. The likelihood of a PPP is given by:

$$L(\lambda|s_1, ..., s_n) = \exp\left(-\int_{\mathcal{S}} \lambda(s)ds\right) \prod_{i=1}^{n} \lambda(s_i). \tag{2.1}$$

2.3.2 Tractability Discussion

The approach adopted thus far in the literature to make nonparametric Bayesian inference on Point processes using Gaussian processes (GPs) consists of putting a *functional prior* on the intensity function in the form of a modulated GP: $\lambda(s) = \phi(g(s))$ where g is drawn from a GP and ϕ is a positive link function. Examples of such ϕ include the exponential function and scaled sigmoid functions (Adams et al. (2009); Rao and Teh (2011)). This approach can be seen as a Cox process where the stochastic intensity follows the same dynamics as the functional prior. When the Gaussian process used has almost surely continuous paths, the random vector

$$\left(\lambda(s_1), ..., \lambda(s_n), \int_{\mathcal{S}} \lambda(s)ds\right) \tag{2.2}$$

provably admits a probability density function (pdf).

It is worth noting that any piece of information not contained in the implied pdf over the vector in Equation (2.2) will be lost as the likelihood only depends on those variables. This observation begs two questions. Is there any real advantage in postulating our prior assumptions as a functional prior on the whole intensity function and *then* deduce the implied marginal over the vector in Equation (2.2)? Assuming

so, does the stochastic process need to be defined through fully dependent marginals like in the case of modulated GP? The answer to the first question is clearly NO for two reasons. First of all, for most useful transformations ϕ and covariance structures for the GP, the implied pdf over the vector in Equation (2.2) might not be available analytically. Second, we do not have to first postulate a functional prior and then derive the aforementioned implied pdf. Crucially, we show in Appendix A.1 that for *any* non-negative supported $(n+1)$-dimensional probability distribution $\mathbb{P}^*$, there exists a stochastic process λ^* indexed on $\mathcal{S}$ with infinitely differentiable and non-negative valued paths, and such that

$$\left(\lambda^*(s_1), ..., \lambda^*(s_n), \int_{\mathcal{S}} \lambda^*(s) ds\right) \sim \mathbb{P}^*. \tag{2.3}$$

In other words, if we choose to directly construct an appropriate prior over the vector in Equation (2.2), we have the guarantee that we would achieve the same result as postulating some functional prior on the intensity function, although the functional prior might not be modulated Gaussian. This observation provides us with room for manoeuvre in the prior specification, and consequently the answer to the second question is also NO. We may for instance choose to introduce an appropriate conditional independence structure in the construction of the prior over the vector in Equation (2.2) in order to scale-up inference.

This approach, which we will adopt in this chapter, departs considerably from the alternative GP approaches of Adams et al. (2009) and Rao and Teh (2011). The cubic complexity of the approach proposed by these authors is simply due to the fact that they chose to put a GP modulated functional prior on the intensity function, and circumvented the need to determine the implied marginal over the vector in Equation (2.2) by using some stability properties of Poisson and renewal processes (namely thinning and uniformization). As previously discussed, this prior construction comes with unnecessary and excessive symmetry assumptions which end up limiting scalability.

Our approach is somewhat similar to that of Kottas (2006). The author regarded

$$I = \int_{\mathcal{S}} \lambda(s) ds$$

as a random variable and noted that

$$p(s) = \frac{\lambda(s)}{\int_{\mathcal{S}} \lambda(s) ds}$$

can be regarded as a pdf whose support is the domain $\mathcal{S}$. He then made inference on

$$(I, p(s_1), ..., p(s_n)),$$

postulating as prior that I and $(p(s_1), ..., p(s_n))$ are independent, I has a Jeffreys prior and $(s_1, ..., s_n)$ are i.i.d. draws from a Dirichlet process mixture of Beta distributions, the pdf of which is p.

The model we present in the following section puts an appropriate *finite-dimensional* prior on $(\lambda(s_1), ..., \lambda(s_n), \lambda(s'_1), ..., \lambda(s'_k), \int_{\mathcal{S}} \lambda(s)ds)$ for some inducing points s'_j rather than putting a *functional prior* on the intensity function directly.

2.3.3 Our Model

Intuition

The intuition behind our model is that the data are not a 'natural grid' at which to infer the value of the intensity function. For instance, if the data consists of 200,000 points on the interval $[0, 24]$ as in one of our experiments, it might not be necessary to infer the value of a function at 200,000 points to characterise it on $[0, 24]$. Instead, we find a small set of inducing points $\mathcal{D}' = \{s'_1, ..., s'_k\}, k \ll n$ on our domain, through which we will define the prior over the vector in Equation (2.2) augmented with $\lambda(s'_1), ..., \lambda(s'_k)$. The set of inducing points will be chosen so that knowing $\lambda(s'_1), ..., \lambda(s'_k)$ would result in knowing the values of the intensity function elsewhere on the domain, in particular $\lambda(s_1), ..., \lambda(s_n)$, with 'arbitrary certainty'. We will then analytically integrate out the dependency in $\lambda(s_1), ..., \lambda(s_n)$ from the posterior, thereby reducing the complexity from cubic to linear in the number of data points without 'loss of information', and reformulating our problem as that of making exact Bayesian inference on the value of the intensity function at the inducing points. We will then describe how to obtain predictive mean and variance of the intensity function elsewhere on the domain from training.

Model Specification

Let us denote by λ^* a positive stochastic process on $\mathcal{S}$ such that $\log \lambda^*$ is a stationary Gaussian process with covariance kernel $\gamma^* : (s_1, s_2) \to \gamma^*(s_1, s_2)$ and constant mean m^*. Let us further denote by $\hat{\lambda}$ a positive stochastic process on $\mathcal{S}$ such that $\log \hat{\lambda}$ is a

conditional Gaussian process coinciding with $\log \lambda^*$ at k inducing points

$$\mathcal{D}' = \{s'_1, ..., s'_k\},\ k \ll n.$$

That is, conditional on λ^*, $\log \hat{\lambda}$ is the nonstationary Gaussian process whose mean function m is defined by

$$m(s) = m^* + \Sigma^*_{s\mathcal{D}'} \Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'} G \tag{2.4}$$

where $G = \left(\log \lambda^*(s'_1) - m^*, ..., \log \lambda^*(s'_k) - m^* \right)$ and Σ^*_{XY} is the covariance matrix between the vectors X and Y under the covariance kernel γ^*. Moreover, $\log \hat{\lambda}$ is such that for every vector S_1 of points in $\mathcal{S}$, the auto-covariance matrix $\Sigma_{S_1 S_1}$ of the values of process at S_1 reads[2]

$$\Sigma_{S_1 S_1} = \Sigma^*_{S_1 S_1} - \Sigma^*_{S_1 \mathcal{D}'} \Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'} \Sigma^{*T}_{S_1 \mathcal{D}'}. \tag{2.5}$$

The prior distribution in our model is constructed as follows:

1. $\{\log \lambda(s'_i)\}_{i=1}^k$ are samples from the stationary GP $\log \lambda^*$ at $\{s'_i\}_{i=1}^k$ respectively, with $m^* = \log \frac{\#\mathcal{D}}{\mu(\mathcal{S})}$, where $\mu(\mathcal{S})$ is the size of the domain.
2. $I = \int_{\mathcal{S}} \lambda(s) ds$ and $\{\log \lambda(s_j)\}_{j=1}^n$ are conditionally independent given $\{\log \lambda(s'_i)\}_{i=1}^k$.
3. Conditional on $\{\log \lambda(s'_i)\}_{i=1}^k$, $\{\log \lambda(s_j)\}_{j=1}^n$ are independent, and for each $j \in [1..n]$, $\log \lambda(s_j)$ follows the same distribution as $\log \hat{\lambda}(s_j)$.
4. Conditional on $\{\log \lambda(s'_i)\}_{i=1}^k$, I follows a gamma distribution with shape α_I and scale β_I.
5. The mean $\mu_I = \alpha_I \beta_I$ and variance $\sigma_I^2 = \alpha_I \beta_I^2$ of I are that of $\int_{\mathcal{S}} \hat{\lambda}(s) ds$.

To ease notations, we define

$$\boldsymbol{\lambda} = (\log \lambda(s_1), ..., \log \lambda(s_n)), \tag{2.6}$$

and

$$\boldsymbol{\lambda}' = (\log \lambda(s'_1), ..., \log \lambda(s'_k)). \tag{2.7}$$

[2]The positive definitiveness of the induced covariance kernel γ is a direct consequence of the positive definitiveness of γ^*.

Assertion 3. above is somewhat similar to the FITC model of Quinonero-Candela and Rasmussen (2005). This construction yields a prior pdf of the form:

$$p\Big(\boldsymbol{\lambda}, \boldsymbol{\lambda}', I, \theta\Big) = \gamma_d\Big(I|\alpha_I, \beta_I\Big)p(\theta)\mathcal{N}\Big(\boldsymbol{\lambda}'|m^*1_k, \Sigma^*_{\mathcal{D}'\mathcal{D}'}\Big)\mathcal{N}\Big(\boldsymbol{\lambda}|M, \mathrm{diag}(\Sigma_{\mathcal{D}\mathcal{D}})\Big) \tag{2.8}$$

where $\mathcal{N}(.|X, C)$ is the multivariate Gaussian pdf with mean X and covariance matrix C, $M = (m(s_1), ..., m(s_n))$, 1_k is the vector with length k and elements 1, $\mathrm{diag}(\Sigma_{\mathcal{D}\mathcal{D}})$ is the diagonal matrix whose diagonal is that of $\Sigma_{\mathcal{D}\mathcal{D}}$, $\gamma_d\,(x|\alpha, \beta)$ is the pdf of the gamma distribution with shape α and scale β, and where θ denotes the hyper-parameters of the covariance kernel γ^*.

It follows from the fifth assertion in our prior specification that $\alpha_I = \frac{\mu_I^2}{\sigma_I^2}$ and $\beta_I = \frac{\sigma_I^2}{\mu_I}$. We also note that

$$\begin{aligned}
\mu_I &= \mathrm{E}\Big(\int_{\mathcal{S}} \hat{\lambda}(s)ds\Big) \\
&= \int_{\mathcal{S}} \mathrm{E}\Big(\exp(\log \hat{\lambda}(s))\Big)ds \\
&= \int_{\mathcal{S}} \exp\left(m(s) + \frac{1}{2}\gamma(s,s)\right)ds \\
&:= \int_{\mathcal{S}} f(s)ds
\end{aligned} \tag{2.9}$$

where we define $f(s) = \exp\left(m(s) + \frac{1}{2}\gamma(s,s)\right)$, and

$$\begin{aligned}
\sigma_I^2 &= \mathrm{E}\left(\Big(\int_{\mathcal{S}} \hat{\lambda}(s)ds\Big)^2\right) - \mu_I^2 \\
&= \mathrm{E}\left(\int_{\mathcal{S}}\int_{\mathcal{S}} \exp(\log \hat{\lambda}(s_1) + \log \hat{\lambda}(s_2))ds_1 ds_2\right) - \mu_I^2 \\
&= \int_{\mathcal{S}}\int_{\mathcal{S}} \mathrm{E}\left(\exp\Big(\log \hat{\lambda}(s_1) + \log \hat{\lambda}(s_2)\Big)\right)ds_1 ds_2 - \mu_I^2 \\
&= \int_{\mathcal{S}}\int_{\mathcal{S}} \exp\Big(m(s_1) + m(s_2) + \gamma(s_1, s_2) + \frac{1}{2}\gamma(s_1, s_1) + \frac{1}{2}\gamma(s_2, s_2)\Big)ds_1 ds_2 - \mu_I^2 \\
&:= \int_{\mathcal{S}}\int_{\mathcal{S}} g(s_1, s_2)ds_1 ds_2 - \mu_I^2,
\end{aligned} \tag{2.10}$$

where we define $g(s_1, s_2) = \exp\Big(m(s_1) + m(s_2) + \gamma(s_1, s_2) + \frac{1}{2}\gamma(s_1, s_1) + \frac{1}{2}\gamma(s_2, s_2)\Big)$. The integrals in Equations (2.9) and (2.10) can be easily evaluated with numerical methods such as Gauss-Legendre quadrature (Hildebrand, 1987, Chap. 8). In particular, when

$\mathcal{S} = [a, b]$,

$$\mu_I \approx \frac{b-a}{2} \sum_{i=1}^{p} \omega_i f\left(\frac{b-a}{2} x_i + \frac{b+a}{2}\right) \tag{2.11}$$

and

$$\sigma_I^2 \approx \frac{(b-a)^2}{4} \sum_{i=1}^{p} \sum_{j=1}^{p} \omega_i \omega_j g\left(\frac{b-a}{2} x_i + \frac{b+a}{2}, \frac{b-a}{2} x_j + \frac{b+a}{2}\right) - \mu_I^2 \tag{2.12}$$

where the roots x_i of the Legendre polynomial of order p and the weights ω_i are readily available from standard textbooks on numerical analysis such as Hildebrand (1987) and scientific programming packages (R, Matlab and Scipy). Extensions to rectangles in higher dimensions are straightforward. Moreover, the time complexity of such approximations only depends on the number of inducing points and the quadrature order p (see Equations (2.4) and (2.5)), and consequently they *scale well* with the data size.

A critical step in the derivation of our model is to analytically integrate out $\log \lambda(s_1), ..., \log \lambda(s_n)$ in the posterior, to eliminate the cubic complexity in the number of data points. To do so, we note that:

$$\begin{aligned} \int \left(\prod_{i=1}^{n} \lambda(s_i)\right) \mathcal{N}\big(\boldsymbol{\lambda} | M, \operatorname{diag}(\Sigma_{\mathcal{D}\mathcal{D}})\big) d\boldsymbol{\lambda} &= \mathrm{E}\left(\exp\left(\sum_{i=1}^{n} \log \lambda(s_i)\right)\right) \\ &= \exp\left(1_n^T M + \frac{1}{2}\operatorname{Tr}(\Sigma_{\mathcal{D}\mathcal{D}})\right), \end{aligned} \tag{2.13}$$

where the second equality results from the moment generating function of a multivariate Gaussian. Thus, putting together the likelihood of Equation (2.1) and Equation (2.8), and integrating out $\boldsymbol{\lambda}$, we get:

$$\begin{aligned} p\big(\boldsymbol{\lambda}', I, \theta \big| \mathcal{D}\big) \;\sim\; & p(\theta) \mathcal{N}\big(\boldsymbol{\lambda}' | M^*, \Sigma^*_{\mathcal{D}'\mathcal{D}'}\big) \exp\left(1_n^T M + \frac{1}{2}\operatorname{Tr}(\Sigma_{\mathcal{D}\mathcal{D}})\right) \\ & \times \exp(-I) \gamma_d\big(I | \alpha_I, \beta_I\big). \end{aligned} \tag{2.14}$$

Finally, although our model allows for joint inference on the intensity function and its integral, we restrict our attention to making inference on the intensity function for brevity. By integrating out I from Equation (2.14), we get the new posterior:

$$p(\boldsymbol{\lambda}', \theta | \mathcal{D}) \sim p(\theta) \exp\left(1_n^T M + \frac{1}{2}\operatorname{Tr}(\Sigma_{\mathcal{D}\mathcal{D}})\right) (1 + \beta_I)^{-\alpha_I} \mathcal{N}\left(\boldsymbol{\lambda}' | M^*, \Sigma^*_{\mathcal{D}'\mathcal{D}'}\right) \tag{2.15}$$

where we noted that the dependencies of Equation (2.14) in I is of the form

$$\exp(-x)\gamma_d(x|\alpha,\beta),$$

which can be integrated out as the moment generating function of the gamma distribution evaluated at -1, that is $(1+\beta)^{-\alpha}$.

Selection of Inducing Points

Inferring the number k and positions of the inducing points s'_i is critical to our model, as k directly affects the complexity of our scheme and the positions of the inducing points affect the quality of our prediction. Too large a k will lead to an unduly large complexity. Too small a k will lead to loss of information (and subsequently excessively uncertain predictions from training), and might make assertion 2 of our prior specification inappropriate. For a given k, if the inducing points are not carefully chosen, the coverage of the domain will not be adapted to changes in the intensity function and as a result, the predictive variance in certain parts of the domain might considerably differ from the posterior variance we would have obtained, had we chosen inducing points in those parts of the domain.

Intuitively, a good algorithm to find inducing points should leverage prior knowledge about the smoothness, periodicity, amplitude and length scale(s) of the intensity function to optimize for the quality of (post-training) predictions while minimising the number of inducing points. We use as utility function for the choice of inducing points:

$$\mathcal{U}(\mathcal{D}') = \mathrm{E}_\theta\left(\mathrm{Tr}\left(\Sigma^*_{\mathcal{D}\mathcal{D}'}(\theta)\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}(\theta)\Sigma^{*T}_{\mathcal{D}\mathcal{D}'}(\theta)\right)\right), \tag{2.16}$$

where θ is the vector of hyper-parameters of the covariance kernel γ^*, and the expectation is taken with respect to the prior distribution over θ. In other words, the utility of a set of inducing points is the expected total reduction of the (predictive) variances of $\log\lambda(s_1),\ldots\log\lambda(s_n)$ resulting from knowing $\log\lambda(s'_1),\ldots,\log\lambda(s'_k)$. The motivation behind this heuristic is to select inducing points $\mathcal{D}'$ such that, as per the prior specification, knowing the values of the intensity function at inducing points would be as 'informative' as possible about the values of the intensity function at data points $\mathcal{D}$. An alternative heuristic would be to minimize the conditional entropy $H\left(\log\lambda(s_1),\ldots\log\lambda(s_n)|\log\lambda(s'_1),\ldots,\log\lambda(s'_k)\right)$. We note however that, because of the conditional independence introduced in our prior specification, this would result in the same inducing points. It is also worth noting that, had it not been for the conditional independence structure we introduced in our prior specification, the conditional entropy

heuristic would have resulted in a time complexity cubic (rather than linear) in the number of data points.

In practice, the expectation in Equation (2.16) might not be available analytically. We can however use as estimate the sample mean

$$\tilde{\mathcal{U}}(\mathcal{D}') = \frac{1}{N}\sum_{i=1}^{N} \operatorname{Tr}\left(\Sigma^{*}_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*T}_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)\right), \tag{2.17}$$

where $\{\tilde{\theta}_i\}_{i\leq N}$ are N i.i.d. draws from the prior. The algorithm proceeds as follows. We sample $\{\tilde{\theta}_i\}_{i\leq N}$ from the prior. Initially we set $k = 0$, $\mathcal{D}' = \varnothing$ and $u_0 = 0$. We increment k by one, and consider adding an inducing point. We then find the point s'_k that maximises $\tilde{\mathcal{U}}(\mathcal{D}' \cup \{s\})$

$$s'_k := \operatorname*{argmax}_{s\in\mathcal{S}} \tilde{\mathcal{U}}(\mathcal{D}' \cup \{s\}) \tag{2.18}$$

using Bayesian optimisation (Močkus (1975, 2012)). We compute the utility of having k inducing points as

$$u_k = \tilde{\mathcal{U}}(\mathcal{D}' \cup \{s'_k\}),$$

we update $\mathcal{D}' = \mathcal{D}' \cup \{s_k\}$ and stop when

$$\frac{u_k - u_{k-1}}{u_k} < \alpha,$$

where $0 < \alpha \ll 1$ is a convergence threshold. This is summarized in Algorithm 2.1. It turns out this algorithm is guaranteed to converge at least as fast as a geometric sequence.

Algorithm 2.1 Selection of inducing points

Inputs: $0 < \alpha \ll 1$, N, p_θ
Output: u_f, $\mathcal{D}'$
$k = 0$, $u_0 = 0$, $\mathcal{D}' = \varnothing$, $e = 1$;
Sample $(\tilde{\theta}_i)_{i=1}^{N}$ from $p(\theta)$;
while $e > \alpha$ **do**
  $k = k + 1$;
  $s'_k = \operatorname*{argmax}_{s\in\mathcal{S}} \tilde{\mathcal{U}}(\mathcal{D}' \cup \{s\})$;
  $u_k = \tilde{\mathcal{U}}(\mathcal{D}' \cup \{s'_k\})$;
  $\mathcal{D}' = \mathcal{D}' \cup \{s'_k\}$;
  $e = \frac{u_k - u_{k-1}}{u_k}$;
end while

Proposition 2.1 *(a) For any $\mathcal{D}$, α, N and p_θ Algorithm 2.1 stops in finite time and the sequence $(u_k)_{k\in\mathbb{N}}$ converges at least linearly with rate $1 - \frac{1}{\#\mathcal{D}}$.*

(b) Moreover, the maximum utility $u_f(\alpha)$ returned by Algorithm 2.1 converges to the average total unconditional variance $w_\infty := \frac{1}{N}\sum_{i=1}^{N} Tr(\Sigma^*_{\mathcal{D}\mathcal{D}}(\tilde{\theta}_i))$ as α goes to 0.

Proof The idea behind the proof of this proposition is that the sequence of maximum utilities u_k is positive, increasing[3], and upper-bounded by the total unconditional variance w_∞[4]. Hence, the sequence u_k converges to a strictly positive limit, which implies that the stopping condition of the while loop will be met in finite time regardless of $\mathcal{D}$, α, N and p_θ. Finally, we construct a sequence w_k upper-bounded by the sequence u_k and that converges linearly to the average total unconditional variance w_∞ with rate $1 - \frac{1}{\#\mathcal{D}}$. As the sequence u_k converges and is itself upper-bounded by w_∞, its limit is w_∞ as well, and it converges at least as fast as w_k. The full proof is provided in Appendix A.2. ■

Our algorithm is particularly suitable to Poisson point processes as it prioritises sampling inducing points in parts of the domain where the data are denser. This corresponds to regions where the intensity function will be higher, thus where the local random counts of the underlying PPP will vary more[5], and subsequently where the posterior variance of the intensity is expected to be higher. Moreover, it leverages prior smoothness assumptions on the intensity function to limit the number of inducing points and to appropriately and sequentially improve coverage of the domain. Algorithm 2.1 is illustrated on a variety of real life and synthetic data sets in section 2.5.

2.4 Inference

We use a *squared exponential* kernel for γ^* and *scaled sigmoid Gaussian* priors for the kernel hyper-parameters; that is

$$\theta_i = \frac{\theta_{i\max}}{1 + \exp(-x_i)} \tag{2.19}$$

where x_i are i.i.d standard normal. The problem-specific scales $\theta_{i\max}$, restrict the supports of those distributions using prior knowledge to avoid unlikely extreme values and to improve conditioning.

[3] Intuitively, conditioning on a new point increases the reduction of variance from the unconditional variance.

[4] The variance cannot be reduced by more than the total unconditional variance.

[5] The variance of the Poisson distribution is its mean.

We use a blocked Gibbs sampler (Geman and Geman (1984)) to sample from the posterior. We sample the hyper-parameters using the Metropolis-Hastings algorithm (Hastings (1970)) taking as proposal distribution the prior of the variable of interest. We sample the log-intensities at the inducing points using Elliptical Slice Sampling (Murray et al. (2010)) with the pdf in Equation (2.15). We could have used Elliptical Slice Sampling (ESS) for sampling the hyper-parameters as well. However, ESS tends to be much costlier than Metropolis-Hastings (MH) per Gibbs cycle, although it typically mixes in fewer Gibbs cycles. Overall, in our experiments we found that using ESS for sampling log-intensities and MH for sampling hyper-parameters mixes faster (in wall-clock time) than the three alternative combinations.

Prediction from Training

To predict the posterior mean at the data points we note from the law of total expectation that

$$\forall s_i \in \mathcal{D},\ \mathrm{E}(\log \lambda(s_i)|\mathcal{D}) = \mathrm{E}\left(\mathrm{E}\left(\log \lambda(s_i)|\{\log \lambda^*(s'_j)\}_{j=1}^k, \mathcal{D}\right)|\mathcal{D}\right). \tag{2.20}$$

Also, we note from Equations (2.1) and (2.8) that the dependency of the posterior of $\log \lambda(s_i)$ conditional on $\{\log \lambda^*(s'_j)\}_{j=1}^k$ is of the form

$$\exp\left(\log \lambda(s_i)\right) \times \mathcal{N}(\log \lambda(s_i)|m(s_i), \gamma(s_i, s_i)),$$

where we recall that $m(s_i)$ is the i-th element of the vector M and $\gamma(s_i, s_i)$ is the i-th diagonal element of the matrix $\Sigma_{\mathcal{D}\mathcal{D}}$. Hence, the posterior distribution of $\log \lambda(s_i)$ conditional on $\{\log \lambda^*(s'_j)\}_{j=1}^k$ is Gaussian with mean

$$\mathrm{E}\left(\log \lambda(s_i)|\{\log \lambda^*(s'_j)\}_{j=1}^k, \mathcal{D}\right) = M[i] + \Sigma_{\mathcal{D}\mathcal{D}}[i, i] \tag{2.21}$$

and variance

$$\mathrm{Var}\left(\log \lambda(s_i)|\{\log \lambda^*(s'_j)\}_{j=1}^k, \mathcal{D}\right) = \Sigma_{\mathcal{D}\mathcal{D}}[i, i]. \tag{2.22}$$

Finally, it follows from Equation (2.20) that $\mathrm{E}(\log \lambda(s_i)|\mathcal{D})$ is obtained by averaging out $M[i] + \Sigma_{\mathcal{D}\mathcal{D}}[i, i]$ over MCMC samples after burn-in. Similarly, the law of total variance implies that

$$\mathrm{Var}(\log \lambda(s_i)|\mathcal{D}) = \mathrm{E}\left(\mathrm{Var}\left(\log \lambda(s_i)|\{\log \lambda^*(s'_j)\}_{j=1}^k, \mathcal{D}\right)|\mathcal{D}\right)$$

$$+ \operatorname{Var}\left(\operatorname{E}\left(\log \lambda(s_i) | \{\log \lambda^*(s'_j)\}_{j=1}^k, \mathcal{D}\right) | \mathcal{D}\right). \tag{2.23}$$

Hence, it follows from Equations (2.21) and (2.22) that the posterior variance at a data point s_i is obtained by summing up the sample mean of $\Sigma_{\mathcal{D}\mathcal{D}}[i,i]$ with the sample variance of $M[i] + \Sigma_{\mathcal{D}\mathcal{D}}[i,i]$, where sample mean and sample variance are taken over MCMC samples after burn-in.

2.5 Experiments

We selected four data sets to illustrate the performance of our model. We restricted ourselves to one synthetic data set for brevity. We chose the most challenging of the synthetic intensity functions of Adams et al. (2009) and Rao and Teh (2011),

$$\lambda(t) = 2\exp\left(-\frac{t}{15}\right) + \exp\left(-\left(\frac{t-25}{10}\right)^2\right),$$

to thoroughly compare our model with competing methods. We also ran our model on a standard 1 dimensional real-life data set (the coal mine disasters dataset used in Jarrett (1979); 191 points) and a standard real-life 2 dimensional data (spatial location of bramble canes from Diggle (1985); 823 points). Finally we ran our model on a real-life data set large enough to cause problems to competing models. This data set consists of the UTC timestamps (expressed in hours in the day) of Twitter updates in English published by Twitter (2014) (on the Twitter sample stream) on September 1st 2014 (188544 points).

2.5.1 Inducing Points Selection

Figure 2.1 illustrates convergence of the selection of inducing points on the 4 data sets. We ran the algorithm 10 times with $N = 20$, and plotted the average normalised utility $\frac{u_k}{u_\infty} \pm 1$ std as a function of the number of inducing points. Table 2.1 contains the maximum hyper-parameters that were used for each data set. Table 2.2 contains the number of inducing points required to achieve some critical normalised utility values for each of the 4 data sets. We note that just 8 inducing points were required to achieve a 95% utility for the Twitter data set (188544 points). In regards to the positions of sampled inducing points, we note from Figures 2.2 and 2.3 that when the intensity function was bimodal, the first inducing point was sampled around the argument of the highest mode, and the second inducing point was sampled around the argument

Table 2.1 Maximum output (resp. input) scale $h_{\max}$ (resp. $l_{\max}$) used for each data set to select inducing points.

	SYNTHETIC	COAL MINE	BRAMBLE	TWITTER
h_{MAX}	10.0	10.0	10.0	10.0
l_{MAX}	25.0	50.0	0.25	5.0

of the second highest mode. More generally, the algorithm sampled inducing points where the latent intensity function varies the most, as expected.

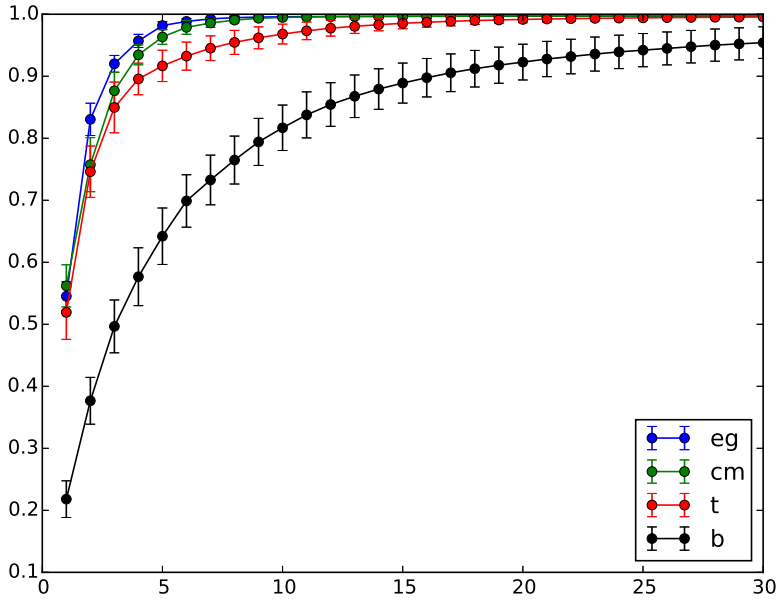

Fig. 2.1 Average normalised utility $\frac{u_k}{u_\infty}$ of choosing k inducing points using Algorithm 2.1 $\pm$ 1 standard deviation as a function of k on the synthetic data set (eg), the coal mine data set (cm), the Twitter data set (t) and the bramble canes data set (b). The average was taken over 10 runs.

2.5.2 Intensity Function

In each experiment we generated 5000 samples after burn-in (1000 samples). For each data set we used the set of inducing points that yielded a 95% normalized utility. The exact numbers are detailed in Table 2.2.

Table 2.2 Number of inducing points produced by Algorithm 2.1 required to achieve a few normalised utility values $\frac{u_k}{u_\infty}$ on the 4 data sets.

	k			
$\frac{u_k}{u_\infty}$	SYNTHETIC	COAL MINE	BRAMBLE	TWITTER
0.75	2	2	8	3
0.90	3	4	17	5
0.95	4	5	28	8

We ran a Monte Carlo simulation for the stochastic processes considered herein and found that the Legendre polynomial order $p = 10$ was sufficient to yield a quadrature estimate for the standard deviation of the integral less than 1% away from the Monte Carlo estimate (using the trapezoidal rule), and a quadrature estimate for the mean of the integral less than a standard error away from the Monte Carlo average. We took a more conservative stand and used $p = 20$.

Inference on Synthetic Data

We generated a draw from a Poisson point process with the intensity function

$$\lambda(t) = 2\exp\left(-\frac{t}{15}\right) + \exp\left(-\left(\frac{t-25}{10}\right)^2\right)$$

of Adams et al. (2009) and Rao and Teh (2011). The draw consisted of 41 points (blue sticks in Figure 2.2). We compared our model to Adams et al. (2009) (SGCP) and Rao and Teh (2011) (RMP). We ran the RMP model with the renewal parameter γ set to 1 (RMP 1), which corresponds to an exponential renewal distribution or equivalently an inhomogeneous Poisson process. We also ran the RMP model with a uniform prior on $[1, 5]$ over the renewal parameter γ (RMP full). Figure 2.2 illustrates the posterior mean intensity function under each model. Finally we ran the Dirichlet Process Mixture of Beta model of Kottas (2006) (DPMB). As detailed in Table 2.3, our model outperformed that of Adams et al. (2009), Rao and Teh (2011) and Kottas (2006) in terms of accuracy and speed.

Although we restricted ourselves to a synthetic dataset for benchmarking, we note that we could have used real datasets for out-of-sample testing. Here we wouldn't be able to draw new held out samples (from the true intensity function) to evaluate the log-predictive likelihood of the learned intensity functions, as we would not know the true intensity function. Moreover, we need to be mindful of the fact that splitting

a dataset into training and testing would effectively change the object of inference (i.e. the intensity function), in such a way that the ground truth intensity functions of both subsets might be completely unrelated to that of the original dataset. One possible approach to circumvent this limitation is to the original dataset $\mathcal{D}$ into a training subset $\mathcal{T}r$ and a testing subset $\mathcal{T}e$ by selecting each point in $\mathcal{D}$ to be part of the training subset $\mathcal{T}r$ with probably $0 < p < 1$, and to be part of the testing subset $\mathcal{T}e$ otherwise. In doing so, it can be shown that, if $\mathcal{D}$ is a random draw from a Poisson point process with intensity function λ, then $\mathcal{T}r$ (resp. $\mathcal{T}e$) is also a draw from a Poisson process with intensity function $p\lambda$ (resp. $(1-p)\lambda$).[6] Thus, the log-predictive likelihood may be evaluated on the testing subset by scaling the intensity function learned during training by $\frac{1-p}{p}$, and an estimate of the intensity function of the original dataset is obtained by scaling the intensity function learned during training by $\frac{1}{p}$.

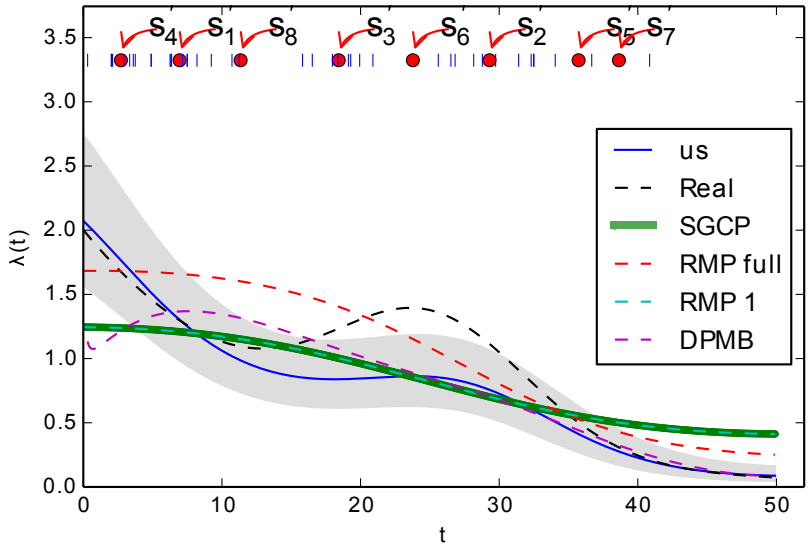

Fig. 2.2 Inference on a draw (blue sticks) from a Poisson point process with intensity $\lambda(t) = 2\exp(-\frac{t}{15}) + \exp(-(\frac{t-25}{10})^2)$ (black line). The red dots are the inducing points generated by our algorithm, labelled in the order they were selected. The solid blue line and the grey shaded area are the posterior mean $\pm$ 1 posterior standard deviation under our model. SGCP is the posterior mean under Adams et al. (2009). RMP full and RMP 1 are the posterior mean intensities under Rao and Teh (2011) with γ inferred and set to 1 respectively. DPMB is the Dirichlet Process mixture of Beta Kottas (2006)

[6] This is a direct application of (Moeller et al., 2003, Prop. 3.6).

Table 2.3 Some statistics for the MCMC runs of Figure 2.2. RMSE and MAE denote the Root Mean Square Error and the Mean Absolute Error, expressed as a proportion of the average of the true intensity function over the domain. LP denotes the log mean predictive probability on 10 held out PPP draws from the true intensity ± 1 std. t(s) is the average time in seconds it took to generate 1000 samples ± 1 std and ESS denotes the average effective sample size (Gelman et al. (2013)) per 1000 samples.

	MAE	RMSE	LP	t (s)	ESS
SGCP	0.31	0.37	-45.07 ± 1.64	257.72 ± 16.29	6
RMP 1	0.32	0.38	-45.24 ± 1.41	110.19 ± 7.37	23
RMP full	0.25	0.31	-43.51 ± 2.15	139.64 ± 5.24	6
DPMB	0.23	0.32	-42.95 ± 3.58	23.27 ± 0.94	**47**
Us	**0.19**	**0.27**	**-42.84 ± 3.07**	**4.35 ± 0.12**	38

Inference on Real-life Data

Figures 2.3, 2.4 and 2.5 show the posterior mean intensity functions of the coal mine data set, the Twitter data set and the bramble canes data set under our model.

We note that it took only 240s on average to generate 1000 samples on the Twitter data set (188544 points). As a comparison, this is the amount of time that would be required to generate as many samples on a data set that has 50 points (resp. 100 points) under the models of Adams et al. (2009) (resp. Rao and Teh (2011)). More importantly, it was not possible to run either of those two competing models on the Twitter data set. Doing so would require computing 17×10^{10} covariance coefficients to evaluate a single auto-covariance matrix of the log-intensity at the data points, which a typical personal computer cannot handle.

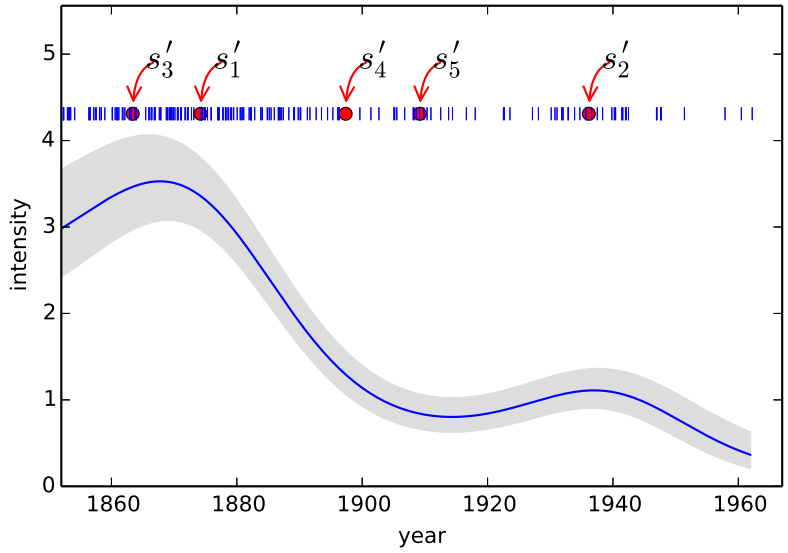

Fig. 2.3 Inference on the intensity functions of the coal mine dataset. Blue dots are data points, red dots are inducing points (labelled in the upper panels in the order they were selected), the grey area is the 1 standard deviation credible band.

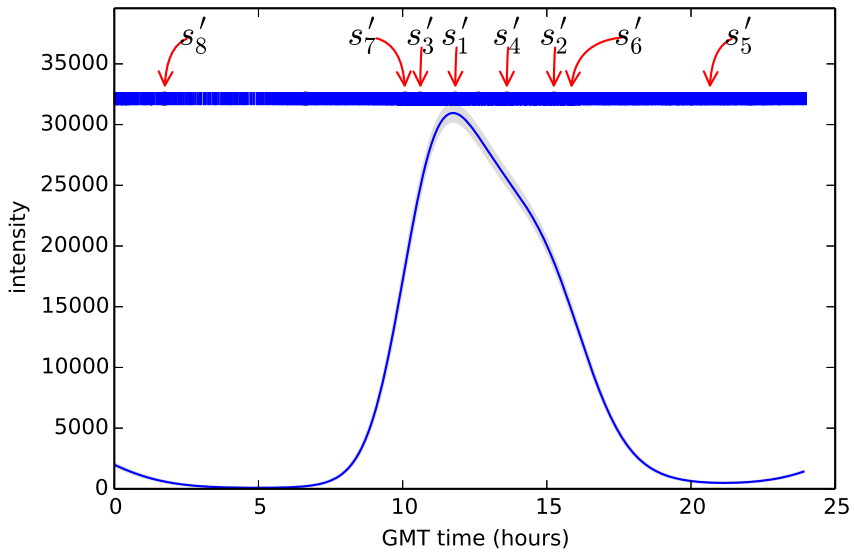

Fig. 2.4 Inference on the intensity functions of the Twitter dataset. Blue dots are data points, red dots are inducing points (labelled in the upper panels in the order they were selected), the grey area is the 1 standard deviation credible band.

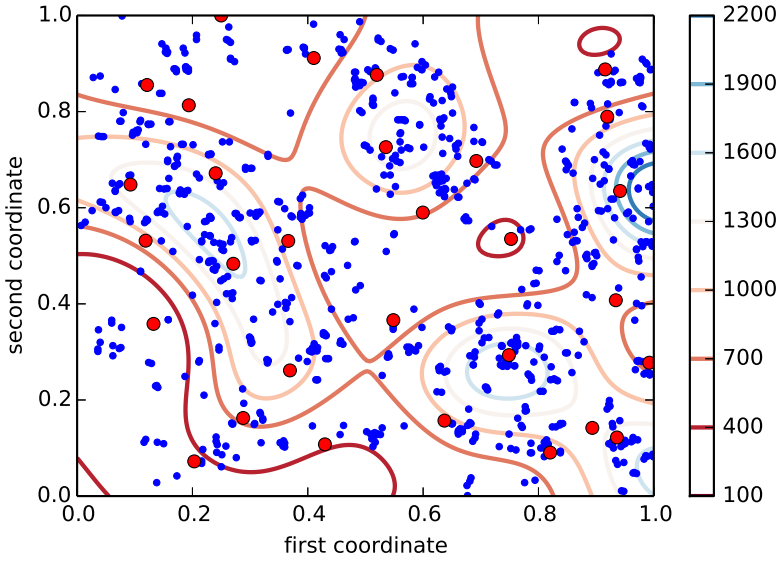

Fig. 2.5 Inference on the intensity functions of the bramble canes dataset. Blue dots are data points (locations of bramble canes), red dots are inducing points selected by our Algorithm 2.1.

2.6 Discussion

2.6.1 Scalability of the Selection of Inducing Points

The computational bottleneck of the inducing point selection is in the evaluation of

$$\mathrm{Tr}\left(\Sigma^*_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*T}_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)\right).$$

Hence, the time complexity and the memory requirement of the inducing point selection are both linear in the number of data points, $n := \#\mathcal{D}$. Moreover, the number of inducing points generated by our algorithm does not increase with the size of the data, but rather as a function of the size of the domain and the resolution implied by the prior over the hyper-parameters.

2.6.2 Comparison with Competing Models

We note that the computational bottleneck of our MCMC inference is in the evaluation of

$$\mathrm{Tr}\left(\Sigma_{\mathcal{D}\mathcal{D}}\right) = \mathrm{Tr}\left(\Sigma^*_{\mathcal{D}\mathcal{D}}\right) - \mathrm{Tr}\left(\Sigma^*_{\mathcal{D}\mathcal{D}'}\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}\Sigma^{*T}_{\mathcal{D}\mathcal{D}'}\right).$$

Hence, inferring the intensity function under our model scales computationally as $\mathcal{O}(nk^2)$ and has a memory requirement $\mathcal{O}(nk)$, where the number of inducing points k is negligible. This is considerably better than alternative methods using Gaussian processes (Adams et al. (2009); Rao and Teh (2011)) whose complexities are cubic in the number of data points and whose memory requirement is squared in the number of data points. Moreover, the superior accuracy of our model, compared to that of Adams et al. (2009) and Rao and Teh (2011), is due to our use of the exponential transformation rather than the scaled sigmoid one. In effect, unlike the inverse scaled sigmoid function that tends to amplify variations, the logarithm tends to smooth out variations. Hence, when the true intensity is uneven, the log-intensity is more likely to resemble a draw from a stationary GP with SE covariance function than the inverse scaled sigmoid of the true intensity function. Consequently, a stationary GP prior with SE covariance function in the inverse domain is more suitable to the exponential transformation than to the scaled sigmoid transformation. Our model is also more suitable than that of Cunningham et al. (2008) when credible bounds are needed for the intensity function, or when the input space is of dimension higher than 1. The model is a useful alternative to that of Kottas (2006), whose complexity is also linear. In effect, Gaussian processes are more flexible than a Dirichlet Process

mixture of Beta distributions. Indeed, unlike a Dirichlet Process mixture of Beta distributions, Gaussian processes allow direct expression of practical prior features such as smoothness, amplitude, length scale(s) (memory), and periodicity.

As our model relies on Gauss-Legendre quadrature, we would not recommend it for applications with a large input space dimension. However, most interesting point process applications involve modelling temporal, spatial or spatio-temporal events, for which our model scales considerably better with the data size than competing approaches. In effect, the models proposed by Cunningham et al. (2007, 2008); Kottas (2006); Rao and Teh (2011) are all specific to unidimensional input data, whereas the model introduced by Kottas and Sansó (2007) is specific to spatial data. The model of Adams et al. (2009) scales poorly with the input space dimension, as its complexity is cubic in the sum of the number of data points and the number of latent thinning points, and the number of thinning points grows exponentially with the input space dimension[7].

2.6.3 Extensions

Although the covariance kernel γ^* was assumed stationary, no result in the proposed approach has relied on that assumption. We solely needed to evaluate covariance matrices under γ^*. Hence, the proposed model and algorithm can also be used to account for known nonstationarities. More generally, the model presented in this paper can serve as foundation to make inference on the stochastic dependency between multiple point processes when the intensities are assumed to be driven by known exogenous factors, hidden common factor, and latent idiosyncratic factors. Additional experiments could explore the effect of the size of the set of induced points on performance, and the degree to which this might correlate with the (greedy) heuristic we used to guide our approach.

2.7 Summary

In this chapter we propose a novel, exact nonparametric model to make inference on Poisson point processes using Gaussian processes. We derive a robust MCMC scheme to sample from the posterior intensity function. Our model outperforms competing benchmarks in terms of speed and accuracy, and it generates less correlated samples

[7]The expected number of thinning points grows proportionally with the volume of the domain, which is exponential in the dimension of the input space when the domain is a hypercube with a given edge length.

(or equivalently higher effective sample sizes) than competing GP based alternatives. A critical advantage of our approach is that it has a numerical complexity and a memory requirement *linear* in the data size n ($\mathcal{O}(nk^2)$, and $\mathcal{O}(nk)$ respectively, with $k \ll n$). Competing models using Gaussian processes have a cubic numerical complexity and squared memory requirement. We show that our model readily handles data sizes not yet considered in the literature. This massive gain in scalability was achieved thanks to a careful construction of a structured prior.

Chapter 3

Online Time Series Forecasting with p-Markov Gaussian Processes

> "To improve is to change; to be perfect is to change often."
>
> *Winston Churchill*

The main objective of this chapter is to illustrate that the lack of scalability in Bayesian kernel methods can be due to postulating excessive smoothness assumptions in the functional prior, and that restricting the assumed degree of differentiability of the latent function to learn can go a long way towards constructing *scalable* and *flexible* kernel methods. To do so, we propose a novel online time series forecasting model that is equivalent to Gaussian process regression under a family of covariance functions, namely *spectral Matérn* kernels, that we prove may perform as well as any oracle stationary covariance function in Gaussian process regression. Moreover, we propose the first scheme for fully-online learning of model parameters in a Gaussian process regression model with one-dimensional inputs, which we apply to our time series forecasting model. Crucially, the overall time complexity and memory requirement of our model for fully-online hyper-parameters learning and fully-online forecasting is constant in the sample size.

3.1 Introduction

Interest in analysing and forecasting time series is thousands of years old. This field of research has witnessed two breakthroughs over the second-half of the twentieth century in the development of the Box-Jenkins methodology (Box et al. (1970)) and

the derivation of the Kalman filter (Kalman (1960)). While the ARIMA model underpinning the Box-Jenkins methodology has since become a standard in graduate curricula worldwide, the Kalman filter has found a great number of applications in addition to time series forecasting, including, but not limited to, object tracking, navigation and computer vision.

3.1.1 Popular Approaches

The Box-Jenkins methodology for analysing time series consists of three steps. Firstly, trends and seasonalities are removed from the training dataset by differentiating the time series iteratively[1] until an ergodic-stationary time series is found. Secondly, the ergodic-stationary time series is assumed to be a realisation of an ARMA process whose parameters are estimated, typically by maximum likelihood. Finally, the calibrated model is checked by analysing the sample autocorrelation of residuals.

Both the ARIMA model and the Kalman filter can be represented as discrete time *linear Gaussian state space models* (see Durbin and Koopman (2012), chap. 3 or Commandeur and Koopman (2007), chap. 10). Discrete time state space models aim at inferring the state $x_k \in \mathbb{R}^p$ of a latent dynamical system, that is assumed to evolve according to Markovian dynamics, from some noisy observations $y_k \in \mathbb{R}^q$. The models are characterised by a probability distribution for the initial latent state of the system $p(x_0)$, a Markovian transition dynamics $p(x_k|x_{k-1})$, and a measurement model $p(y_k|x_k)$. It is also assumed that the observations are independent from each other and from other latent states, conditional on their associated latent states (i.e. $\forall i \neq j \; y_i \perp y_j|x_i$ and $y_i \perp x_j|x_i$). The forecasting problem then consists of recursively determining the conditional distribution

$$p(x_{k+n}|y_1, \ldots, y_k)$$

for some $n > 0$.

In linear Gaussian state space models (LGSSM), the three distributions above are taken to be Gaussian, and the transition and measurement distributions have covariance matrices that do not depend on the state process. Moreover, it is assumed that

$$\mathrm{E}\left(x_k|x_{k-1}\right) = F_k x_{k-1} + B_k u_k, \quad \mathrm{E}\left(y_k|x_k\right) = H_k x_k,$$

[1] That is the following time series are constructed from the original time series y_k: $\Delta^{(1)} y_k := y_k - y_{k-1}$, $\ldots$, $\Delta^{(i+1)} y_k := \Delta^{(i)} y_k - \Delta^{(i)} y_{k-1}$ etc...

where u_k is a control-vector, and the control-input model B_k, the state transition (or plant) model F_k and the observation model H_k are matrices that are given (or estimated off-line). Under this class of models the forecasting problem can be solved exactly and very efficiently using the Kalman filter equations.

LGSSMs can be extended in several ways. Alternatives have been proposed, including the popular *extended Kalman filter* and *unscented Kalman filter*, that relax the linearity assumption in LGSSMs by postulating that

$$\mathrm{E}\left(x_k|x_{k-1}\right) = f(x_{k-1}, u_k), \ \mathrm{E}\left(y_k|x_k\right) = h(x_k),$$

for some (possibly non-linear) functions f and h, while retaining the Gaussian assumptions on the initial, transition and measurement distributions. Other approaches extend LGSSMs by relaxing the foregoing Gaussian assumptions. For instance, discrete measurement probability distributions such as the Poisson distribution or the negative binomial distribution would allow for count (as opposed to real-valued) observations (West et al. (1985)), while leptokurtic probability distributions such as the Student-t distribution or the Laplace distribution, used as measurement distribution, might improve the robustness of forecasts to outliers. For a more comprehensive review of Box-Jenkins models and state space methods for time series analysis, we refer the reader to Commandeur and Koopman (2007), Durbin and Koopman (2012) and Hamilton (1994).

More generally, time series forecasting can be regarded as a regression problem, where one is interested in predicting future values and associated credible bounds from historical observations. Gaussian process regression or GPR (Rasmussen and Williams, 2005, chap. 2) provides a flexible Bayesian nonparametric framework for that purpose. The GPR approach to time series modelling consists of regarding the time series as arising from the sum of a continuous time Gaussian process $(z_t)_{t\geq 0}$ with mean function m and covariance function k_θ, and independent Gaussian white noise:

$$\forall t_i \geq 0, \ y_{t_i} = z_{t_i} + \epsilon_{t_i},$$

$$(z_t)_{t\geq 0} \sim \mathcal{GP}\left(m(.), k_\theta(.,.)\right), \ (\epsilon_t)_{t\geq 0} \sim \mathcal{GWN}(\sigma^2).$$

As a result, $(y_t)_{t\geq 0}$ is also a Gaussian process (GP), and the noise variance σ^2 and all other hyper-parameters can easily be inferred from some historical data $\{y_{t_0}, \ldots, y_{t_k}\}$ by maximising the corresponding multivariate Gaussian likelihood. Predictive distributions may then be obtained in closed-form.

More recently, online *passive-aggressive* algorithms have been proposed by Crammer et al. (2006), that allow for online linear and basis function regression with strong worst-case loss bounds.

3.1.2 Limitations of Popular Approaches

The critical assumption underpinning the Box-Jenkins methodology is that any time series, after enough iterative differentiations, will become *ergodic-stationary*, or more precisely will pass standard ergodic-stationarity statistical tests. The fundamental problem with this approach is that in practice, one will have access to finite samples that might not be sufficiently large or informative for the time series to pass any ergodic-stationarity test. It is in this spirit that Durbin and Koopman (2012), §3.10.1 noted that 'in the economic and social fields, real series are never stationary however much differencing is done'. The ergodic-stationarity assumption is not a major limitation per se, but relying on the assumption that sufficient data have been collected to largely characterise the latent process (including testing for ergodic-stationarity) is a major limitation of the Box-Jenkins methodology. In that regards, Kwiatkowski et al. (1992) noted that 'most economic time series are not very informative about whether or not there is a unit root'. The state space approach on the other hand does not require that the sample be informative enough to test for stationarity, which is why Commandeur and Koopman (2007) and Durbin and Koopman (2012) argued it should be preferred to the Box-Jenkins methodology.

Additionally, the scalability of state space models is very appealing. However, the dynamics of the latent time series, largely characterised by F_k (f in the non-linear case) is usually assumed to be known. Although this assumption is fairly mild in many engineering applications where the dynamics are derived from the laws of physics, when analysing time series arising from complex systems where the dynamics are not known, F_k (resp. f) fully characterize the conditional covariance function of the latent time series $(x_t)_{t\geq 0}$. Thus, hand-picking F_k or f would be problematic as the covariance function of the latent process should be flexibly learned from the data to appropriately uncover and exploit patterns. Unfortunately, no state space dynamics has been proposed in the literature, to the best of our knowledge, that guarantees that the implied covariance functions of the latent processes $(x_t)_{t\geq 0}$ can perform as well as any stationary covariance function.

As for GPR, with a cubic time complexity and a squared memory requirement, it scales poorly with the number of training samples. Sparse approximations have been proposed that scale linearly with the number of observations (Lazaro-Gredilla et al.

(2010); Quinonero-Candela and Rasmussen (2005)). However, those methods are not good enough for streaming applications as they require loss of information through windowing to be of practical interest.

Passive-aggressive (PA) algorithms (Crammer et al. (2006)) have constant time complexity and memory requirement for online linear regression problems. However, their kernelized versions have memory requirement and computational time that depend on the number of 'support vectors', and might increase unboundedly with the number of observations (Wang and Vucetic (2010)). Thus, PA algorithms in their original form are not appropriate for flexibly forecasting time series in a high throughput setting. Wang and Vucetic (2010) have introduced PA algorithms with an additional memory or CPU 'budget' constraint, but the authors restricted themselves to classification problems. Moreover, unlike probabilistic approaches, PA algorithms do not provide uncertainty around predictions. A Bayesian PA-like online learning approach has been proposed by Shi and Zhu (2014). However, like all other kernel-PA algorithms, the kernel is assumed to be given and online learning of its parameters is not addressed by the authors.

3.1.3 Our Contributions

In this chapter, we propose a family of stationary kernels we refer to as *spectral Matérn* kernels that we prove can perform as well as any oracle stationary covariance function on any Gaussian process regression task, while independently enabling the learning or controlling of the degree of differentiability of the latent Gaussian process. We propose a state space model for analysing and forecasting time series that we prove is equivalent to GPR under a *spectral Matérn* covariance function. More importantly, we derive a novel scheme for online learning of hyper-parameters as a solution to a convex optimization problem, which happens to be a *passive-aggressive* algorithm (Crammer et al. (2006)) with *online gradient descent* updates (Bottou (1998)). Overall, our model has constant time complexity and constant memory requirement, both for online learning of the hyper-parameters and for online prediction.

The rest of the chapter is structured as follows. In section 3.2 we provide the intuition behind our approach and the related mathematical background. In section 3.3 we formally introduce our model. We illustrate that our approach outperforms competing approaches with some experiments in section 3.4. Finally, we discuss possible extensions and conclude in section 3.5.

3.2 Intuition and Background

In this chapter we consider modelling real-valued time series. Our working assumption is that samples $\{y_{t_0}, \ldots, y_{t_N}\}$ of a possibly asynchronous time series arise as noisy observations of a 'smooth'[2] latent stochastic process $(z_t)_{t\geq 0}$ perturbed by an independent white noise process $(\epsilon_t)_{t\geq 0}$:

$$\forall t_i \geq 0,\ y_{t_i} = z_{t_i} + \epsilon_{t_i}, \tag{3.1}$$

$$(\epsilon_t)_{t\geq 0} \sim \mathcal{WN}(\sigma^2),\ (\epsilon_t)_{t\geq 0} \perp (z_t)_{t\geq 0}. \tag{3.2}$$

In financial econometrics applications for instance, the process $(z_t)_{t\geq 0}$ may represent the logarithm of the equilibrium or fair price of an asset, while $(\epsilon_t)_{t\geq 0}$ accounts for short term shocks, microstructure noise and so-called 'fat-fingers'. More generally, $(z_t)_{t\geq 0}$ denotes the value of the time series of interest, after taking out hazards such as side effects, measurement noise, recording errors and so forth, accounted for by $(\epsilon_t)_{t\geq 0}$.

Moreover, we will assume that the latent process $(z_t)_{t\geq 0}$ is a trend-stationary Gaussian process, i.e. a Gaussian process with a translation-invariant covariance function and an arbitrary mean function:

$$(z_t)_{t\geq 0} \sim \mathcal{GP}(m(.), k(.)). \tag{3.3}$$

Allowing for a non-constant mean function is crucial in applications where one would expect the phenomenon of interest to exhibit a long-term trend, such as global population growth and world GDP per capita to name but a few. This is due to the fact that, by Slutsky's theorem (see Papoulis and Pillai, 2002, §13.1, Eq. (13-10)), a sufficient condition for the centred stationary process $(z^c_t)_{t\geq 0}$, with $z^c_t := z_t - \mathrm{E}(z_t)$, to be mean-ergodic and consequently mean-reverting, is that

$$k(\tau) \underset{\tau\to+\infty}{\longrightarrow} 0. \tag{3.4}$$

The foregoing condition is desirable in our model as it can be regarded as a *finite memory* assumption on the latent process, which will be met by most time series of practical interest. Thus restricting the mean function of the latent process $(z_t)_{t\geq 0}$ to be constant would imply postulating that the latent process asymptotically oscillates about a constant. It is our working assumption that the class of trend-stationary Gaussian processes is large enough for our purpose. We refer the reader to Appendix

[2] By smooth we mean that the process is at least mean square continuous, and higher order derivatives might exist.

B.1 for a discussion on why this is a relatively mild assumption. From a Bayesian perspective, we are assuming *a priori* that the latent process smoothly evolves about a deterministic trend, and that our *prior* uncertainty on how the process might deviate from the trend is invariant by translation. In applications, the trend m will typically be taken to lie in a parametric family of functions.

Furthermore, we assume that the white noise process $(\epsilon_t)_{t\geq 0}$ is Gaussian:

$$(\epsilon_t)_{t\geq 0} \sim \mathcal{GWN}(\sigma^2). \tag{3.5}$$

We will discuss possible extensions to more robust noise models in section 3.5. Our setup thus far is the same as GPR with a translation-invariant covariance function. Our next objective is to investigate whether we could exploit the scalability of the state space approach by providing an 'equivalent' state space representation. More precisely, by 'equivalent' we mean one in which the observable is the noisy time series value $y_t \in \mathbb{R}$, and for every collection of observation times $\{t_0, \ldots, t_N\}$, the joint distribution over $(y_{t_0}, \ldots, y_{t_N})$ as per the state space representation is the same as that implied by Equations (3.1) to (3.5). Thus, we need to construct an appropriate Markov process

$$(x_t)_{t\geq 0},\ x_t \in \mathbb{R}^p,$$

that is related to $(z_t)_{t\geq 0}$ and from which the state space dynamics will be deduced. We will then derive the measurement dynamics from Equation (3.1). As $(x_t)_{t\geq 0}$ should be a Markov process, intuitively x_t should contain all information about present and past values of the latent time series $(z_t)_{t\geq 0}$ if we want the state space representation to be equivalent to Equations (3.1) to (3.5). Thus, it seems natural to consider that $(z_t)_{t\geq 0}$ is a coordinate process of the vector-valued state process $(x_t)_{t\geq 0}$.

Case 1: $\forall t \geq 0,\ x_t = z_t$
If we take the state process to be the latent time series itself, $(z_t)_{t\geq 0}$ then needs to be Markovian. However, this would imply that the latent time series will either not be mean square differentiable, or, as the following lemma states, it will almost surely be equal to the trend plus a constant, which might be too restrictive.

Lemma 3.1 *If a real-valued trend-stationary Gaussian process with differentiable mean function is mean square differentiable and Markovian, then it has a constant covariance function (or equivalently it is almost surely equal to its mean function plus a constant).*

Proof See Appendix B.2. ■

Intuitively, the memorylessness of the Markov assumption conflicts with the memoryfulness of the differentiability assumption, so that the choice $x_t = z_t$ might only be appropriate when assuming that the smooth latent time series $(z_t)_{t\geq 0}$ is mean square continuous but not differentiable. An example covariance function yielding a continuous trend-stationary Markov Gaussian process is the Matérn-$\frac{1}{2}$ (also called exponential) kernel k_{exp} (see Rasmussen and Williams, 2005, Appendix §B.2). The equivalent state space representation is easily derived from standard Gaussian identities as:

$$\begin{cases} z_{t_0} \sim \mathcal{N}\left(m(t_0), k_{\text{exp}}(0)\right), \\ z_{t_k} = m(t_k) + \alpha_k \left(z_{t_{k-1}} - m(t_{k-1})\right) + \sqrt{k_{\text{exp}}(0)\left(1-\alpha_k^2\right)}\xi_{t_k} \\ y_{t_k} = z_{t_k} + \epsilon_{t_k} \end{cases}$$

where

$$\alpha_k = \frac{k_{\text{exp}}(t_k, t_{k-1})}{k_{\text{exp}}(0)},$$

and

$$(\xi_t)_{t\geq 0} \sim \mathcal{GWN}(1).$$

Case 2: $\forall t \geq 0,\ x_t = \left(z_t, z_t^{(1)}, \ldots, z_t^{(p)}\right)$

In many applications, we might be interested in postulating that the latent time series $(z_t)_{t\geq 0}$ is smoother, for instance p times mean square differentiable. In this case, we derive our intuition for what might be good candidates to complete the state process into an appropriate Markov vector-valued process from the theory of analytic functions underpinning implementations of scientific functions in most programming languages. Conceptually, the aforementioned theory implies that under mild conditions, the value of an infinitely smooth function and its derivatives $f(t_k), f^{(1)}(t_k), \ldots, f^{(i)}(t_k), \ldots$ at a given point t_k are sufficient to fully characterise the entire function elsewhere on the domain. Although we only require $(z_t)_{t\geq 0}$ to be p times differentiable, it is our hope that by augmenting the state process with the p consecutive derivatives of $(z_t)_{t\geq 0}$, we could ensure that (i) each state variable x_t contains all information about past values of the latent time series $(x_u)_{u\leq t}$ or equivalently the state process $(x_t)_{t\geq 0}$ is Markovian, (ii) the latent time series $(z_t)_{t\geq 0}$ is as smooth as desired, and (iii) the covariance function of the latent time series is not strongly restricted.

Definition 3.2 *(p-derivative Gaussian process) When $(z_t)_{t\geq 0}$ is a Gaussian process that is p times continuously differentiable in the mean square sense, we denote the p-derivative Gaussian process (pDGP) as the vector-valued Gaussian process $(x_t)_{t\geq 0}$*

where $x_t = \left(z_t, z_t^{(1)}, \ldots, z_t^{(p)}\right)$ *and* $(z^{(i)})_{t\geq 0}$ *is the* i*-th order mean square derivative of* $(z_t)_{t\geq 0}$.

Definition 3.3 *(p-Markov Gaussian process) We say that* $(z_t)_{t\geq 0}$ *is a p-Markov Gaussian process (pM-GP) when it is p times continuously differentiable and the corresponding pDGP is Markovian.*[3]

Stationary pM-GPs have been extensively studied in the seminal paper by Doob (1944).[4] In particular, the fundamental theorem below (see Doob, 1944, Theorem 4.9 (ii)) provides necessary and sufficient conditions on our covariance function k for the augmented state process to be Markovian when the trend m is constant.

Theorem 3.4 *A real-valued stationary Gaussian process* $(z_t)_{t\geq 0}$ *with integrable covariance function* k *is p-Markov if and only if* k *admits a spectral density of the form*

$$\forall \omega > 0,\ S(\omega) := \int k(\tau) e^{-2\pi i \omega \tau} d\tau = \frac{c}{A(\omega^2)}, \tag{3.6}$$

where $c > 0$ *and* A *is a polynomial of degree* $p+1$.

Corollary 3.5 *A stationary Gaussian process with covariance function given by the Matérn kernel*

$$k_{MA}(\tau; k_0, l, \nu) = k_0 \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}|\tau|}{l}\right)^{\nu} K_\nu \left(\frac{\sqrt{2\nu}|\tau|}{l}\right), \tag{3.7}$$

where $k_0 > 0$, $l > 0$, Γ *is the Gamma function,* K_ν *is the modified Bessel function of second kind, and with* $\nu = p + \frac{1}{2}$, $p \in \mathbb{N}$, *is p-Markov.*

Proof The spectral density of a Matérn-$(p+\frac{1}{2})$ covariance function is of the form

$$\frac{c}{(\alpha + 4\pi^2\omega^2)^{p+1}}$$

with $c, \alpha > 0$ (see Rasmussen and Williams, 2005, Appendix §B.2). ■

Before deriving the state space representation of trend-stationary GPs with Matérn-$(p+\frac{1}{2})$ covariance functions, we recall that a pDGP is a vector-valued Gaussian process and that in the trend-stationary case

$$\forall\ 0 \leq i, j \leq p,\ \operatorname{cov}\left(z_u^{(i)}, z_v^{(j)}\right) = (-1)^j k^{(i+j)}(u - v),$$

[3] A 0M-GP ($p = 0$) is a Gaussian Markov process.
[4] In Doob's paper a pM-GP is denoted a t.h.G.M.$_{(p-1)}$ process.

where we use the superscript $^{(i)}$ to denote the i-th order derivative or the original function (or process) when $i = 0$ (see Adler and Taylor, 2011, §2.6). If we further denote

$$K_{u,v} := \left\{ \mathrm{cov}\left(z_u^{(i)}, z_v^{(j)} \right) \right\}_{0 \leq i,j \leq p}$$

the covariance matrix between x_u and x_v, as

$$K_{u|v} := K_{u,u} - K_{u,v} K_{v,v}^{-1} K_{v,u}$$

the auto-covariance matrix of x_u conditional on x_v, and $L_{u|v}$ as any square matrix satisfying[5]

$$K_{u|v} = L_{u|v} L_{u|v}^T,$$

then we obtain the following equivalent LGSSM representation (see Appendix B.5 for details):

$$\begin{cases} x_{t_0} \sim \mathcal{N}(0, K_{t_0,t_0}) \\ x_{t_k} = F_{t_k} x_{t_{k-1}} + L_{t_k|t_{k-1}} \xi_{t_k} \\ y_{t_k} = m(t_k) + H^T x_{t_k} + \epsilon_{t_k} \end{cases} \tag{3.8}$$

with $F_{t_k} = K_{t_k,t_{k-1}} K_{t_{k-1},t_{k-1}}^{-1}$, $H = (1, 0, \ldots, 0)$, and where $(\xi_t)_{t \geq 0}$ is now a $(p+1)$-dimensional standard Gaussian white noise process.

Approaches have been proposed to construct state-space approximations to GPR models (Särkkä et al. (2013); Solin and Särkkä (2014)). These approaches suffer from many pitfalls. First of all the user is expected to i) choose a stationary covariance function that is appropriate for the time series of interest *beforehand*, ii) manually derive the spectral density of the chosen covariance function when it is not straightforward and iii) manually approximate the spectral density of the chosen covariance function by a rational function. In an age of automation, this is not suitable: the model should automatically learn the covariance function from the data, without any adverse effect on scalability. The second pitfall is that model parameters are learned *offline* with linear time complexity. In real-life however the characteristics of the underlying dynamical system are often expected to change, for instance in financial time series, and the forecasting model should automatically cope with these changes *online*, and with resource requirement that do not increase with the sample size in order to enable streaming applications.

The solutions to the pitfalls described above that we propose in this chapter consist of i) introducing a family of stationary kernels, namely *spectral Matérn* kernels, that

[5] The Cholesky factorisation and the Singular Value Decomposition provide such a decomposition.

provably can perform as well as any oracle kernel in GPR and that allow for an *exact* state-space representation, and ii) deriving a fully-online scheme for learning all model parameters with time complexity and memory requirement that do not depend on the sample size. The state space representation of our time series model (Equations (3.1) to (3.5)) under a spectral Matérn covariance function that we will derive is inspired by the state space representation of p-Markov Matérn Gaussian processes (Equation (3.8)). The forecasting problem will be solved exactly using standard Kalman filter and Gaussian process techniques. As for the algorithm to learn model parameters online, it will be derived as a solution to a convex optimization problem, and it will happen to be a *passive-aggressive* algorithm with online gradient descent updates.

3.3 Our Model

The discussion on flexible stationary kernels we provide in the following subsection is an extract of Chapters 6 and 7, where we provide a broader analysis, beyond stationarity and beyond Gaussian process regression.

3.3.1 Flexible Choice of Covariance Function

We start by introducing spectral Matérn kernels, which we prove are as good as any competing stationary covariance function in GPR problems.

It doesn't seem far-fetched to think that for a family of stationary kernels to be flexible enough it should at least contain kernels that may approximate any continuous stationary kernel up to an arbitrarily small precision. In order to construct such a family, we need a characterisation of continuous stationary kernels, one of which is provided by Bochner's theorem.

Theorem 3.6 *(Bochner's theorem) A real-valued continuous function k on $\mathbb{R}^d$ is a stationary covariance function if and only if it can be represented as*

$$k(\tau) = \int_{\mathbb{R}^d} \cos\left(2\pi\omega^T\tau\right)\mu(d\omega), \tag{3.9}$$

where μ is a positive finite measure.

Bochner's theorem introduces a duality between the class of continuous stationary covariance functions and the class of positive finite measures μ in the spectral domain. The characterisation it provides is well complemented by the notion of weak convergence of measures (Bergström (2014)) which we recall below.

Definition 3.7 *(Weak convergence of measures) A sequence of measures μ_n defined on $\left(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d)\right)$ weakly converges to a measure μ on $\left(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d)\right)$ when for all continuous bounded functions f,*

$$\int_{\mathbb{R}^d} f(\omega)\mu_n(d\omega) \underset{n\to+\infty}{\longrightarrow} \int_{\mathbb{R}^d} f(\omega)\mu(d\omega). \tag{3.10}$$

In particular, it immediately follows from the boundedness of the cosine function in Equation (3.9) that, if a sequence of positive finite measures μ_n weakly converges to another positive finite measure μ, then for every $\tau \in \mathbb{R}^d$ the sequence

$$k_n(\tau) := \int_{\mathbb{R}^d} \cos\left(2\pi\omega^T\tau\right)\mu_n(d\omega)$$

converges to

$$k(\tau) := \int_{\mathbb{R}^d} \cos\left(2\pi\omega^T\tau\right)\mu(d\omega).$$

In other words, a sufficient condition for a family of continuous real-valued stationary kernels $\{\dots, k_n, \dots\}$ to be pointwise dense in the family of all real-valued continuous stationary kernels[6], or equivalently to contain kernels that may (pointwise) approximate any continuous stationary kernel arbitrarily well, is that the corresponding family of spectral measures $\{\dots, \mu_n, \dots\}$ be weakly dense in the family of all positive finite measures. Moreover, it is well-known that positive finite measures of the form

$$\mu_{\text{sing.}} = \sum_{i=0}^{n} k_{0i}\delta_{\{\omega_i\}},$$

where $\omega_i \in \mathbb{R}^d,\ k_{0i} \geq 0,\ n \in \mathbb{N}$, and

$$\forall A \subset \mathbb{R}^d, \delta_{\{x\}}(A) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases},$$

are weakly dense in the space of all positive finite measures on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ (Hu and Papageorgiou, 2013, lemma 2.9). Hence, the corresponding family of stationary kernels

$$\left\{k_{\text{SS}}(\tau; n) : \omega_i \in \mathbb{R}^d,\ k_{0i} \geq 0,\ 0 \leq i \leq n,\ n \in \mathbb{N}\right\}, \tag{3.11}$$

with

$$k_{\text{SS}}(\tau; n) := \sum_{i=0}^{n} k_{0i} \cos\left(2\pi\omega_i^T\tau\right), \tag{3.12}$$

[6] A family of functions $\mathcal{F}$ is said to be pointwise dense in a family of function $\mathcal{G}$ when for all $g \in \mathcal{G}$ there is a sequence of functions $f_n \in \mathcal{F}$ that converges pointwise to g.

is pointwise dense in the family of all continuous stationary covariance functions. In particular the following proposition holds.

Proposition 3.8 *For any $\nu > 0$, the family of kernels we refer to as spectral Matérn kernels, which we define as*

$$\{k_{SMA}(\tau; \nu, n) : \omega_i \in \mathbb{R},\ k_{0i}, l_i \geq 0,\ n \in \mathbb{N}\}, \tag{3.13}$$

with

$$k_{SMA}(\tau; \nu, n) := \sum_{i=0}^{n} k_{MA}(\tau; k_{0i}, l_i, \nu) \cos(2\pi\omega_i \tau), \tag{3.14}$$

where k_{MA} is the Matérn kernel defined in Equation (3.7), is pointwise dense in the family of continuous stationary covariance functions defined on $\mathbb{R}$.

Proof See Appendix B.3. ■

It turns out that pointwise density in the space of continuous stationary covariance functions is a guarantee of performance relative to any other continuous stationary kernel in GPR problems as stated below.

Proposition 3.9 *Let $p(\mathcal{D}|k^*, m)$ denote the marginal likelihood in a GPR problem with training data $\mathcal{D}$, mean function m, and covariance function k^*. Then for any continuous stationary kernel k, and for any $\nu > 0$, there exists a sequence of spectral Matérn kernels $\{k_{SMA}(\tau; \nu, n)\}_{n\in\mathbb{N}}$ such that*

$$p\left(\mathcal{D}|k_{SMA}(.; \nu, n), m\right) \underset{n\to+\infty}{\longrightarrow} p(\mathcal{D}|k, m).$$

Proof See Appendix B.4. ■

In practical terms, Proposition 3.9 means that for any oracle continuous stationary kernel k, we may always find a spectral Matérn kernel whose model evidence on the same GPR problem is equal to that of k up to an arbitrarily small precision. In other words, to the extent that the model evidence is used for model selection, spectral Matérn kernels are guaranteed to be as good as any alternative hand-crafted stationary kernel.

Remark: It is worth stressing that the above result holds for any $\nu > 0$. Noting that when $\nu = p + \frac{1}{2},\ p \in \mathbb{N}^*$ (resp. when $\nu = \frac{1}{2}$) spectral Matérn kernels give rise to p times mean square continuously differentiable (resp. mean square continuous) stationary

GPs, it follows that spectral Matérn kernels enable the learning or controlling of the degree of differentiability of the latent function, independently from the requirement for flexibility. In contrast, the approach of Särkkä et al. (2013) cannot simultaneously approximate the target covariance function and the degree of differentiability of the latent GP. In order to approximate a stationary GP with a state space representation, the authors approximate its covariance function with one that has as spectral density a rational function of the form

$$S(\omega) = R(\omega^2) = \frac{P(\omega^2)}{Q(\omega^2)}, \tag{3.15}$$

where P and Q are polynomials with no common zero and degrees p and q respectively, $p < q$. Clearly, for such an approximation to be flexible enough, q has to be large. Indeed, the number of symmetric pairs of local extrema of S is the number of local extrema of R, and R provably has less than $q(q-1)$ local extrema. Considering that a spectral Matérn kernel with n spectral components has spectral density with up to n local extrema[7], when n is high, q needs to be high otherwise the rational function approximation of its spectral density would provably be poor. However, a stationary GP whose covariance function has spectral density of the form of Equation (3.15) is $(q-1)$ times mean square continuously differentiable. Consequently, a spectral Matérn covariance function with high n and $\nu = \frac{1}{2}$ cannot be properly approximated by the method of Särkkä et al. (2013): either the degree of regularity of the latent GP (namely mean square continuity) is honoured (i.e. $q = 1$) in which case the approximation of the kernel would be poor[8], or the approximation of the function is improved with a high q, in which case the approximating GP might be considerably smoother than desired, which might impact predictive accuracy.

In the next section we address this limitation of Särkkä et al. (2013) by providing an alternative state space model that independently allows controlling the degree of regularity of the latent GP and learning of the covariance function.

[7]Its spectral density arises as linear combination of n translations of the spectral density of the corresponding Matérn, and the latter is known to have a single extremum at 0.

[8]This would be equivalent to approximating the spectral Matérn kernel with a simple exponential kernel.

3.3.2 The p-Markov Gaussian Process Filter

Next, we introduce a state space time series model we refer to as the p-Markov Gaussian process filter (pM-GP filter) that we prove is equivalent to Gaussian process regression under a *spectral Matérn* covariance function.

Definition 3.10 *(pM-GP filter) We denote p-Markov Gaussian process filter (pM-GP filter) as a state space model of the form:*

$$\begin{cases} \perp \left\{ {}^i_c x_{t_0}, {}^i_s x_{t_0} \right\}_{i=0}^n, \perp \left\{ ({}^i_c \xi_t)_{t\geq 0}, ({}^i_s \xi_t)_{t\geq 0} \right\}_{i=0}^n \\ \forall i,\ {}^i_c x_{t_{k-1}} \perp {}^i_c \xi_{t_k},\ {}^i_s x_{t_{k-1}} \perp {}^i_s \xi_{t_k} \\ \forall i,\ {}^i_c x_{t_k} \perp \epsilon_{t_k},\ {}^i_s x_{t_k} \perp \epsilon_{t_k} \\ \forall i,\ {}^i_c x_{t_0}, {}^i_s x_{t_0} \sim \mathcal{N}(0, {}^i K_{t_0,t_0}) \\ \forall i,\ {}^i_c x_{t_k} = {}^i F_{t_k} {}^i_c x_{t_{k-1}} + {}^i L_{t_k|t_{k-1}} {}^i_c \xi_{t_k} \\ \forall i,\ {}^i_s x_{t_k} = {}^i F_{t_k} {}^i_s x_{t_{k-1}} + {}^i L_{t_k|t_{k-1}} {}^i_s \xi_{t_k} \\ y_{t_k} = m(t_k) + H^T \sum_{i=0}^n \left(\cos(\omega_i t_k) {}^i_c x_{t_k} + \sin(\omega_i t_k) {}^i_s x_{t_k} \right) + \epsilon_{t_k} \end{cases},$$

where $\perp$ denotes mutual independence, $({}^i_c \xi_t)_{t\geq 0}$ and $({}^i_s \xi_t)_{t\geq 0}$ are $(p+1)$-dimensional standard Gaussian white noise processes, $(\epsilon_t)_{t\geq 0}$ is a scalar Gaussian white noise with variance σ^2, ${}^i K_{u,v}$ is the cross-covariance matrix of a pDGP with Matérn kernel $k_{MA}(\tau; k_{0i}, l_i, p+\frac{1}{2})$, and where $H, {}^i F_{t_k}, {}^i L_{t_k|t_{k-1}}$ are as in the previous section (replacing $K_{u,v}$ by ${}^i K_{u,v}$).

The following proposition establishes that the pM-GP filter is equivalent to Gaussian process regression (GPR) under a spectral Matérn kernel, and a possibly non-constant mean function.

Proposition 3.11 *Let $(\hat{z}_t)_{t\geq 0}$ be a trend-stationary Gaussian process with mean function m and spectral Matérn covariance function*

$$k_{SMA}(\tau) = \sum_{i=0}^{n} k_{MA}\left(\tau; k_{0i}, l_i, p + \frac{1}{2}\right) \cos(\omega_i \tau). \tag{3.16}$$

Let $(\hat{\epsilon}_t)_{t\geq 0}$ be Gaussian white noise with variance σ^2, that is independent from $(\hat{z}_t)_{t\geq 0}$, and $(\hat{y}_t)_{t\geq 0}$ the process defined as

$$\forall t \geq 0,\ \hat{y}_t = \hat{z}_t + \hat{\epsilon}_t.$$

Finally, let $(y_t)_{t\geq 0}$ be the observation process of the pM-GP filter with the same parameters $m, n, p, \sigma, \{k_{0i}, l_i, \omega_i\}_{i=0}^n$. Then, the processes $(y_t)_{t\geq 0}$ and $(\hat{y}_t)_{t\geq 0}$ have the same law, or equivalently:

$$\forall t_0 < \cdots < t_N,\ (y_{t_0}, \ldots, y_{t_N}) \sim (\hat{y}_{t_0}, \ldots, \hat{y}_{t_N}).$$

Proof See Appendix B.6. ■

3.3.3 Solution to the Forecasting Problem

We note that the pM-GP filter can be rewritten as

$$\begin{cases} \boldsymbol{x}_{t_0} \sim \mathcal{N}(0, \boldsymbol{K}_{t_0,t_0}) \\ \boldsymbol{x}_{t_k} = \boldsymbol{F}_{t_k} \boldsymbol{x}_{t_{k-1}} + \boldsymbol{L}_{t_k|t_{k-1}} \boldsymbol{\xi}_{t_k} \\ y_{t_k} = m(t_k) + \boldsymbol{H}_{t_k}^T \boldsymbol{x}_{t_k} + \epsilon_{t_k} \end{cases}. \tag{3.17}$$

Here, $\boldsymbol{x}_{t_k}$ (resp. $\boldsymbol{\xi}_{t_k}$) is the $2(p+1)(n+1)$ vector obtained by stacking up the the vectors $\{\ldots, {}_c^i x_{t_k}, {}_s^i x_{t_k}, \ldots\}$ (resp. $\{\ldots, {}_c^i \xi_{t_k}, {}_s^i \xi_{t_k}, \ldots\}$). Similarly, $\boldsymbol{K}_{t_0,t_0}$ (resp. $\boldsymbol{F}_{t_k}, \boldsymbol{L}_{t_k|t_{k-1}}, \boldsymbol{K}_{t_k|t_{k-1}}$) is the $2(p+1)(n+1) \times 2(p+1)(n+1)$ block diagonal matrix whose $2i$-th and $(2i+1)$-th diagonal blocks are both ${}^i K_{t_0,t_0}$ (resp. ${}^i F_{t_k}, {}^i L_{t_k|t_{k-1}}, {}^i K_{t_k|t_{k-1}}$). Finally, $\boldsymbol{H}_{t_k}$ is the $2(p+1)(n+1)$-dimensional vector with coordinates 0 except at indices which are multiples of $p+1$, and

$$\forall i \in [0 \ldots n],\ \boldsymbol{H}_{t_k}[2i(p+1)] = \cos(\omega_i t_k)$$

and

$$\boldsymbol{H}_{t_k}[(2i+1)(p+1)] = \sin(\omega_i t_k).$$

Thus, the pM-GP filter is a LGSSM, and the forecasting problem can be solved exactly, iteratively and in closed-form, and with memory requirement and time complexity both constant in the total number of observations. Using standard Kalman

filter and Gaussian processes techniques (see Appendix B.7) we obtain:

$$\begin{cases} \forall t > t_{k-1},\ \boldsymbol{x}_t | y_{t_0:t_{k-1}} & \sim \mathcal{N}(\boldsymbol{m}_t^-, \boldsymbol{P}_t^-) \\ \forall t > t_{k-1},\ y_t | y_{t_0:t_{k-1}} & \sim \mathcal{N}(m(t) + \boldsymbol{H}_t^T \boldsymbol{m}_t^-, v_t^-) \\ \forall t > t_{k-1},\ z_t | y_{t_0:t_{k-1}} & \sim \mathcal{N}(m(t) + \boldsymbol{H}_t^T \boldsymbol{m}_t^-, v_t^- - \sigma^2) \\ \forall t \geq t_{k-1},\ \boldsymbol{x}_t | y_{t_0:t} & \sim \mathcal{N}(\boldsymbol{m}_t, \boldsymbol{P}_t) \\ \forall t \geq t_{k-1},\ z_t | y_{t_0:t} & \sim \mathcal{N}(m(t) + \boldsymbol{H}_t^T \boldsymbol{m}_t, v_t) \end{cases} \tag{3.18}$$

with

$$y_{t_0:t_{k-1}} = \{y_{t_0}, \ldots, y_{t_{k-1}}\},$$
$$z_t = m(t) + \boldsymbol{H}_t^T \boldsymbol{x}_t,$$

Prediction Step:

$$\boldsymbol{m}_{t_0}^- = 0,\ \boldsymbol{P}_{t_0}^- = \boldsymbol{K}_{t_0,t_0} \tag{3.19}$$
$$\forall k \geq 1, t > t_{k-1},\ \begin{cases} \boldsymbol{m}_t^- & = \boldsymbol{F}_t \boldsymbol{m}_{t_{k-1}} \\ \boldsymbol{P}_t^- & = \boldsymbol{F}_t \boldsymbol{P}_{t_{k-1}} \boldsymbol{F}_t^T + \boldsymbol{K}_{t|t_{k-1}} \end{cases},$$

and
Update Step:

$$\forall t \geq t_0,\ \begin{cases} v_t^- & = \boldsymbol{H}_t^T \boldsymbol{P}_t^- \boldsymbol{H}_t + \sigma^2 \\ \boldsymbol{e}_t^- & = y_t - m(t) - \boldsymbol{H}_t^T \boldsymbol{m}_t^- \\ \boldsymbol{G}_t & = \frac{1}{v_t^-} \boldsymbol{P}_t^- \boldsymbol{H}_t \\ \boldsymbol{m}_t & = \boldsymbol{m}_t^- + \boldsymbol{e}_t^- \boldsymbol{G}_t \\ \boldsymbol{P}_t & = \boldsymbol{P}_t^- - v_t^- \boldsymbol{G}_t \boldsymbol{G}_t^T \\ v_t & = \boldsymbol{H}_t^T \boldsymbol{P}_t \boldsymbol{H}_t \end{cases}. \tag{3.20}$$

3.3.4 Online Learning of Hyper-Parameters

Noting that $y_{t_0} \sim \mathcal{N}(m(t_0), v_{t_0}^-)$, and that

$$\log p(y_{t_0:t_N}) = \log p(y_{t_0}) + \sum_{k=1}^{T} \log p(y_{t_k} | y_{t_0:t_{k-1}}),$$

it follows that maximum likelihood inference of the hyper-parameters of m, σ, and $\{k_{0i}, l_i, \omega_i\}_{i=0}^n$ may be achieved with linear time complexity $\mathcal{O}(T)$ and with constant memory requirement using Equations (3.18, 3.19, 3.20). However, to be of practical interest in streaming applications, this time complexity requires a loss of information through windowing, which might be detrimental to forecasting performance. We herein propose an alternative online approach that has constant time complexity.

So far we have assumed that the hyper-parameters are constant over time. However, if we extend the model to iteration-specific hyper-parameters, the solution to the forecasting problem (Equations (3.18, 3.19, 3.20)) will provably remain unchanged. The intuition behind our approach to online learning of the hyper-parameters is borrowed from maximum likelihood inference when the hyper-parameters are constant, and from online passive-aggressive algorithms for regression, classification and semi-supervised learning (Chang et al. (2010); Crammer et al. (2006); Wang and Vucetic (2010)). Let $\boldsymbol{\theta}_{t_k}$ be the vector of hyper-parameters (or log hyper-parameters for positive hyper-parameters) prevailing at time t_k. We recall that when hyper-parameters are constant, we may write

$$\log p_{\boldsymbol{\theta}}\left(y_{t_0:t_T}\right) = \log p_{\boldsymbol{\theta}}\left(y_{t_0}\right) + \sum_{k=1}^{T} \log p_{\boldsymbol{\theta}}\left(y_{t_k} | y_{t_0:t_{k-1}}\right).$$

Let us define

$$\mathcal{L}_{t_k}^l(\boldsymbol{\theta}) := \begin{cases} \log p_{\boldsymbol{\theta}}\left(y_{t_0}\right) & \text{if } k = 0 \\ \log p_{\boldsymbol{\theta}}\left(y_{t_k} | y_{t_0:t_{k-1}}\right) & \text{if } k \neq 0 \end{cases}.$$

At time t_k, $\mathcal{L}_{t_k}^l(\boldsymbol{\theta})$ represents the (log) likelihood that $\boldsymbol{\theta}$ explains the new observation y_{t_k} given all previous observations (and hyper-parameters) and thus can be regarded as a *local utility* function that we should aim to maximize. However, we should also be mindful of the 'utility' of $\boldsymbol{\theta}$ in accounting for all previously observed data $(y_{t_0}, \ldots, y_{t_{k-1}})$, which can be achieved by ensuring that the new update is not too far away from $\boldsymbol{\theta}_{t_{k-1}}$. Deriving online learning updates as solutions to a constrained optimization problem that provides a trade-off between retaining information from previous iterations and locally reducing a loss function, has been advocated for a long time in the online learning literature (Chang et al. (2010); Crammer et al. (2006); Helmbold et al. (1999); Kivinen and Warmuth (1997); Littlestone (1989); Wang and Vucetic (2010)). The approach we adopt is largely inspired by the PA-II algorithm of Crammer et al. (2006).

We first consider the problem:

$$\begin{cases} \min\limits_{\boldsymbol{\theta},\xi} \; ||\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}}||^2 + c_k\xi^2 \\ \text{s.t.} \;\; \max\left(-\epsilon - \mathcal{L}^l_{t_k}(\boldsymbol{\theta}), 0\right) \leq \xi \end{cases}, \tag{3.21}$$

where $\epsilon \geq 0$ and $c_k > 0$. c_k can be regarded as an aggressiveness parameter: the larger c_k the more weight is given to increasing the local utility compared to retaining information from previous iterations. On the other hand ϵ is a margin or tolerance parameter that allows for some degree of local sup-optimality, thereby helping to avoid overfitting. Although this problem provides a suitable trade-off, it is tedious to solve analytically, and it might not have a closed-form solution. However, noting that the objective function of problem (3.21) aims at minimising $||\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}}||$, we may replace $\mathcal{L}^l_{t_k}(\boldsymbol{\theta})$ with its first order Taylor expansion at $\boldsymbol{\theta}_{t_{k-1}}$. We adopt the resulting problem for online learning of the hyper-parameters:

$$\begin{cases} \boldsymbol{\theta}_{t_k}, \xi_{t_k} = \underset{\boldsymbol{\theta},\xi}{\text{argmin}} \; ||\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}}||^2 + c_k\xi^2 \\ \text{s.t.} \;\; \max\left(-\epsilon - \hat{\mathcal{L}}^l_{t_k}(\boldsymbol{\theta}), 0\right) \leq \xi \end{cases}, \tag{3.22}$$

with $\hat{\mathcal{L}}^l_{t_k}(\boldsymbol{\theta}) = \mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}) + \nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})^T(\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}})$. The above optimization problem is convex and has closed-form solution (see Appendix B.8 for the derivation, and Appendix B.9 for the derivation of $\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta})$):

$$\boldsymbol{\theta}_{t_k} = \boldsymbol{\theta}_{t_{k-1}} + c_k \frac{\max\left(-\epsilon - \mathcal{L}^l_{t_k}\left(\boldsymbol{\theta}_{t_{k-1}}\right), 0\right)}{1 + c_k\|\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})\|^2} \nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}). \tag{3.23}$$

Noting that the parameters do not change when $\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}) > -\epsilon$, it follows that our learning scheme can be regarded as a *passive-aggressive* algorithm with *online gradient descent* (OGD) update.

Our approach has two main advantages over OGD. Firstly, it is much more robust to the choice of aggressiveness parameter (or learning rate). Indeed, our update is equivalent to

$$\boldsymbol{\theta}_{t_k} = \boldsymbol{\theta}_{t_{k-1}} + c_k \max\left(-\epsilon - \mathcal{L}^l_{t_k}\left(\boldsymbol{\theta}_{t_{k-1}}\right), 0\right) \nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})$$

Algorithm 3.1 Forecasting with the pM-GP filter.

In: $\{y_{t_k}\}_{k\geq 0}$, average sampling frequency F_s.
Out: Predictive probability density functions $\{\dots, p(z_t|y_{t_0}:y_{t_k}), \dots\}, t \geq t_k$.

Set $\omega_i = \frac{1+i}{1+n}\pi F_s$, set the other parameters to 0.
for all (t_k, y_{t_k}) **do**
  Evaluate $\boldsymbol{\theta}_{t_k}$ using Equation (3.23).
  Run the prediction step Eqs. (3.19) with $t = t_k$.
  Run the update step Eqs. (3.20) with $t = t_k$.
  For $t \geq t_k$ get $p(z_t|y_{t_0}:y_{t_k})$ from Eqs. (3.18, 3.19).
end for

for very small c_k, and to

$$\boldsymbol{\theta}_{t_k} = \boldsymbol{\theta}_{t_{k-1}} + \frac{\max\left(-\epsilon - \mathcal{L}^l_{t_k}\left(\boldsymbol{\theta}_{t_{k-1}}\right), 0\right)}{\|\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})\|^2}\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})$$

for very large c_k. In contrast, OGD explodes as the learning rate c_k increases. The second advantage is that our PA-approach guards against overfitting unlike OGD. Indeed, OGD would update parameters at each step, whereas in our PA-approach, whether parameters should be updated or not, and by how much they should be updated, depends on the extent to which the local utility they yield is worse than the tolerance threshold ϵ — this is due to the one-sided ϵ-insensitive loss term

$$\max\left(-\epsilon - \mathcal{L}^l_{t_k}\left(\boldsymbol{\theta}_{t_{k-1}}\right), 0\right).$$

It is also worth noting that, given that

$$\hat{\mathcal{L}}^l_{t_k}(\boldsymbol{\theta}_{t_k}) - \hat{\mathcal{L}}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}) = \max\left(-\epsilon - \mathcal{L}^l_{t_k}\left(\boldsymbol{\theta}_{t_{k-1}}\right), 0\right)\frac{c_k\|\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})\|^2}{1 + c_k\|\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})\|^2} \geq 0,$$

each change in the hyper-parameters in our approach increases the approximate conditional log likelihood $\hat{\mathcal{L}}^l_{t_k}$.

Algorithm 3.1 summarizes online forecasting and hyper-parameters learning under the pM-GP filter.

In order to control the aggressiveness of our learning algorithm in a manner that is consistent across datasets, we rewrite the aggressiveness parameter in the form

$$c_k := c\frac{\|\boldsymbol{\theta}_{t_{k-1}}\|^2}{\left(\epsilon + \hat{\mathcal{L}}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})\right)^2},\ c > 0,$$

that arises by normalising the term $||\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}}||^2$ (resp. ξ^2) in the objective function of problem (3.22) by $\|\boldsymbol{\theta}_{t_{k-1}}\|^2$ (resp. $\left(\epsilon + \hat{\mathcal{L}}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})\right)^2$).

3.4 Experiments

In this section we start by demonstrating that the pM-GP filter provides considerably more accurate forecasts than competing fully-online approaches on standard real-life time series datasets. We then experimentally illustrate the sensitivity of the forecasting errors to the normalized aggressiveness parameter c, the trend, and the number of spectral components n.

3.4.1 Benchmarking

We compare our model to competing fully-online alternatives on the CO_2 dataset of Rasmussen and Williams (2005), and the airline passengers dataset of Box et al. (1970). We select as competing benchmarks three autoregressive (AR) models, namely AR(1), AR(2) and AR(10), and we use four different algorithms to learn the autoregressive coefficients online, namely the PA, PA-I and PA-II algorithms of Crammer et al. (2006), and Bayesian online linear regression with i.i.d. standard normal priors on the weights (BLR). For consistency with the pM-GP filter, we initialize the autoregressive weights in all four algorithms to 0. We choose $c = 100.0, \epsilon = 0.0$ for pM-GP models and PA algorithms. We choose a linear trend and twice differentiability ($p = 2$) for the pM-GP filter. BLR is run with a noise standard deviation of 5% of the sample standard deviation of the corresponding time series. For each model we perform one-step ahead forecast. Mean absolute errors[9] normalized by the standard deviation of increments (NMAE) are reported in Table 3.1, where we refer to all three PA algorithms as PAs because they perform identically up to two decimal points. The evolutions of the running NMAE as a function of time are illustrated in Figure 3.1 and 3.2 for both datasets. We note that our approach provides *considerably* more accurate forecasts than competing alternatives.

3.4.2 Sensitivity to Parameters

The sensitivity of the accuracy of the pM-GP filter to the trend, the aggressiveness parameter c and the number of spectral components is illustrated in Figure 3.3 and

[9]Excluding the first forecast, which is the same for all models.

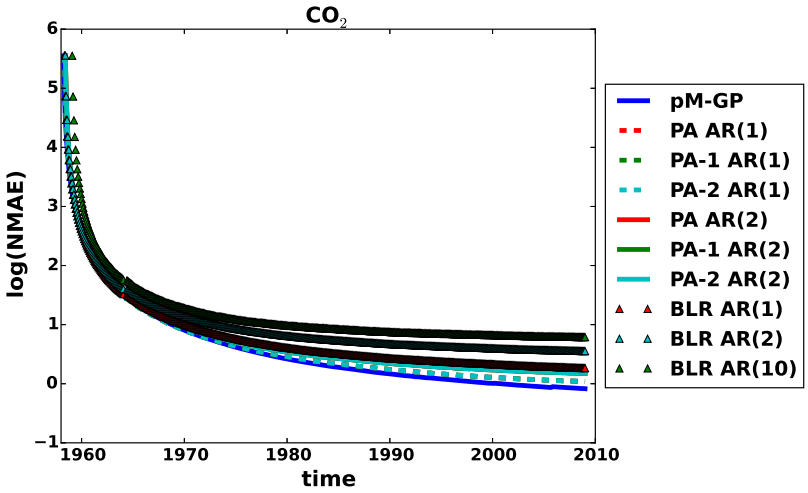

Fig. 3.1 Comparison of various online time series forecasting models on the CO_2 dataset of Rasmussen and Williams (2005). The figure depicts the running (normalized) mean absolute error (NMAE) as a function of time. Absolute errors are normalized by the standard deviation of increments of the time series. The models considered are a twice differentiable pM-GP filter (i.e. $p = 2$), and auto-regressive models of order 1, 2 and 10 whose parameters are learned through Bayesian linear regression (BLR), or passive-aggressive algorithms (PA, PA-1, PA-2).

Figure 3.4 for the CO_2 dataset. Overall, for our data size (607 points), it can be seen that the error decreases as a function of c. This is in line with the empirical observation of Crammer et al. (2006) (Figure 5) that, for small datasets large c perform better, and $c = 0.001$ begins to outperform $c = 100.0$ for the PA classification problems the author considered when the data size is in the thousands. Moreover, we note from Figure 3.4 that for small datasets, a large n should not necessarily be preferred to a smaller one, as more samples will typically be required to learn the model hyper-parameters.

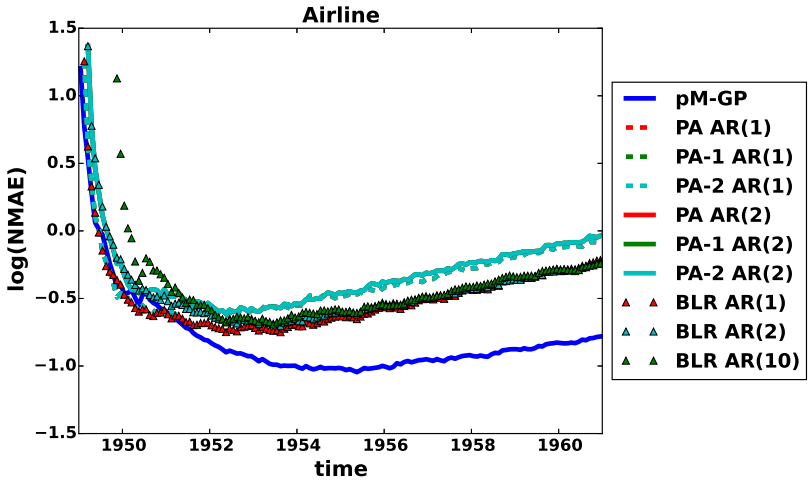

Fig. 3.2 Comparison of various online time series forecasting models on the airline passengers dataset of Box et al. (1970). The figure depicts the running (normalized) mean absolute error (NMAE) as a function of time. Absolute errors are normalized by the standard deviation of increments of the time series. The models considered are a twice differentiable pM-GP filter (i.e. $p = 2$), and auto-regressive models of order 1, 2 and 10 whose parameters are learned through Bayesian linear regression (BLR), or passive-aggressive algorithms (PA, PA-1, PA-2).

Table 3.1 Mean ± 1 standard error of normalized absolute errors in the online time series forecasting experiments of section 3.4. Absolute errors are normalized by the standard deviation of increments of the time series. The models considered are a twice differentiable pM-GP filter (i.e. $p = 2$), and auto-regressive models of order 1, 2 and 10 whose parameters are learned through Bayesian linear regression (BLR), or passive-aggressive algorithms (PAs).

	CO_2	Airline
pM-GP	**0.49 ±0.02**	**0.44 ± 0.03**
PAs AR(1)	0.61 ± 0.02	0.92 ± 0.06
BLR AR(1)	0.88 ± 0.02	0.79 ± 0.05
PAs AR(2)	0.77 ± 0.02	0.94 ± 0.07
BLR AR(2)	1.31 ± 0.02	0.77 ± 0.05
PAs AR(10)	1.02 ± 0.02	0.92 ± 0.07
BLR AR(10)	1.77 ± 0.04	0.77 ± 0.05

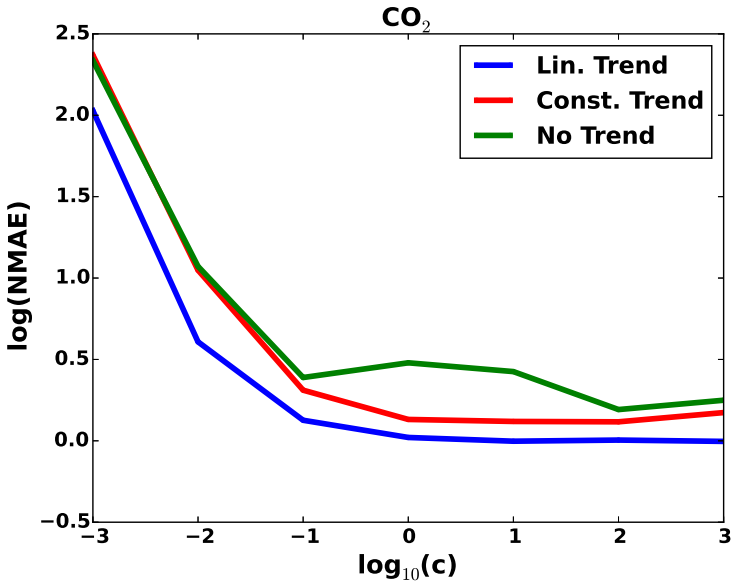

Fig. 3.3 Effect of the pM-GP normalized aggressiveness parameter c on NMAE on the CO_2 experiment.

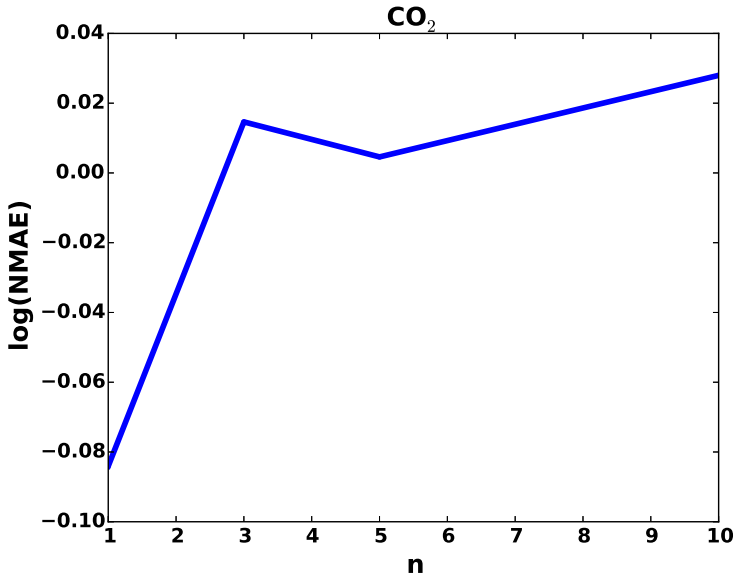

Fig. 3.4 Effect of the number of spectral components n on NMAE on the CO_2 experiment.

3.5 Discussion

3.5.1 Extensions

The approach may easily be extended to structured time series prediction, thus allowing for online learning of correlations between time series. Techniques that extend the Kalman filter to leptokurtic measurement noise models (e.g. Agamennoni et al. (2011)) may also be used out-of-the box to robustify our approach. Further work could also attempt to give bounds for the performance of the online parameter learning method.

3.5.2 Finite Smoothness is Good for Scalability

Thus far we have illustrated that when the latent process is restricted to be at most p times differentiable, we may perform *fully-online*, *scalable* and *flexible* Bayesian time series forecasting. What about the reverse? Can we develop an equivalent LGSSM representation of our time series model (Equations (3.1) to (3.5)) when the latent process $(z_t)_{t\geq 0}$ is assumed to be infinitely differentiable — or equivalently when the covariance function is infinitely differentiable?

We briefly touched on the case where the (Markov) state process is unidimensional in section 3.2, and we showed that if the latent (Markov) state process is mean square differentiable and stationary, then it is almost surely constant (see Lemma 3.1). This result can be generalized to the case where the latent (Markov) Gaussian state process $(x_t)_{t\geq 0}$ is $\mathbb{R}^q$-valued with $q > 1$. In that case, it was shown by Doob (see Doob, 1944, Theorem 4.6 v) that if $(x_t)_{t\geq 0}$ is mean square differentiable (i.e. each of its q coordinate processes is mean square differentiable), then the coordinate processes of $(x_t)_{t\geq 0}$ are infinitely differentiable, and each coordinate process is deterministic and a solution to a q-th order homogeneous linear differential equation. In other words, if the $\mathbb{R}^q$-valued Gaussian process $(x_t)_{t\geq 0}$ is mean square differentiable and Markov, then it is fully characterized by a finite number of initial conditions, and so is the latent smooth process

$$z_t := H_t^T x_t, \tag{3.24}$$

where $t \to H_t$ is an $\mathbb{R}^q$-valued deterministic function. This case would only be useful when $(z_t)_{t\geq 0}$ is assumed to follow a specific deterministic differential equation, with unknown initial conditions, which is rather restrictive for some domain such as high frequency finance. Reciprocally, if we assume that a stationary Gaussian process $(z_t)_{t\geq 0}$ is infinitely differentiable with *strictly positive definite* covariance function and may be

decomposed as

$$z_t := H_t^T x_t, \tag{3.25}$$

where $t \to H_t$ is an $\mathbb{R}^q$-valued infinitely differentiable deterministic function, then $(x_t)_{t\geq 0}$ cannot be a Markov Gaussian process. If it was, then we could always find n sample $(z_{t_0}, \ldots, z_{t_n})$ that are linearly dependent by virtue of the previous discussion, which would be a contradiction of the fact that $(z_t)_{t\geq 0}$ has a *strictly positive definite* covariance function. Consequently, the following result holds true.

Proposition 3.12 *No Gaussian process regression model on a unidimensional input space, and with stationary,* ***infinitely differentiable*** *and strictly positive definite covariance function admits an equivalent LGSSM representation.*

Moreover, noting that our approach has memory requirement and time complexity that grow quadratically and cubically with the degree of differentiability of the latent smooth time series, it follows that a finite smoothness assumption is good for scalability.[10] That being said, in most practical applications, there is little practical advantage in postulating smoothness assumptions beyond twice continuous differentiability (Rasmussen and Williams (2005)). Future works could help empirically confirm how the choice of the degree of differentiability affects accuracy and efficiency.

3.6 Summary

We propose an exact state space representation of Gaussian process regression for time series forecasting, under a family of kernels that is dense in the family of *all* stationary kernels, and that allows decoupling the flexibility of the covariance structure from the differentiability requirement of the latent time series. When hyper-parameters are known, exact GP predictive inference can be performed in constant time and with constant memory requirement. Critically, we propose a novel *passive-aggressive* algorithm for online learning of the model hyper-parameters. The overall approach we refer to as the pM-GP filter has constant time complexity and memory requirement, and provides more accurate forecasts than fully-online competing alternatives on standard datasets such as CO_2 dataset and airline passengers dataset.

[10] Some scalable kernel methods have been proposed that work best with smooth kernels (e.g. Nyström's method). However, these methods are usually *ex-post* approximations that might not have been needed in the first place, had the problem exhibited a suitable conditional independence structure, like LGSSMs.

If there is one single takeaway we want the reader to carry over into the next chapter, it is that in Bayesian kernel methods, an excessive smoothness assumption about the latent function of interest can negatively impact scalability.

Chapter 4

String and Membrane Gaussian Processes

> "It is the long history of humankind (and animal kind, too) those who learned to collaborate and improvise most effectively have prevailed."
>
> *Charles Darwin*

In Chapters 2 and 3 we illustrated that the lack of scalability of Bayesian kernel methods often results from the lack of conditional independence structure in the finite dimensional marginals of the stochastic process used as functional prior and/or excessive smoothness assumptions, and we have proposed application-specific solutions. In this chapter we provide a generalization. We introduce a novel class of stochastic processes that, when used as functional prior in any Bayesian kernel method, allows for *scalable* and *flexible* inference of latent functions in the presence of *locally homogeneous patterns*. The core idea underpinning the contribution of this chapter is to go back to basics (namely the Kolmogorov extension theorem) and incorporate suitable conditional independence structures directly in the construction of the proposed stochastic processes, which will allow us to address the needs for flexibility and scalability at the same time, without resorting to approximations, while ensuring a suitable degree of global smoothness.[1]

[1] The contributions in Chapters 4 and 5 have been published in the Journal of Machine Learning Research.

4.1 Introduction

Many problems in statistics and machine learning involve inferring a latent function from training data (for instance regression, classification, inverse reinforcement learning, inference on point processes to name but a few). Real-valued stochastic processes, among which Gaussian processes (GPs), are often used as functional priors for such problems, thereby allowing for a full Bayesian nonparametric treatment. In the machine learning community, interest in GPs grew out of the observation that some Bayesian neural networks converge to GPs as the number of hidden units approaches infinity (Neal (1996)). Since then, other similarities have been established between GPs and popular models such as Bayesian linear regression, Bayesian basis function regression, splines models and support vector machines (Rasmussen and Williams (2005)). However, they often perform poorly on *Big Data* tasks primarily for two reasons. Firstly, large datasets are likely to exhibit multiple types of local patterns that should appropriately be accounted for by flexible and possibly nonstationary covariance functions, the development of which is still an active subject of research. Secondly, inference under GP priors often consists of looking at the values of the GP at all input points as a jointly Gaussian vector with fully dependent coordinates, which induces a memory requirement and time complexity respectively squared and cubic in the training data size, and thus is intractable for large datasets. We refer to this approach as the *standard GP paradigm*. The framework we introduce in this chapter addresses both of the above limitations.

The contributions of this chapter are as follows. We introduce a novel class of stochastic processes, string Gaussian processes (*string GPs*), that may be used as priors over latent functions within a Bayesian nonparametric framework, especially for large scale problems and in the presence of (possibly multiple types of) local patterns. We propose a framework for analysing the flexibility of random functions and surfaces, and prove that our approach yields more flexible stochastic processes than isotropic Gaussian processes. We demonstrate that exact inference under a *string GP* prior has the potential of scaling considerably better than in the *standard GP paradigm*, and is amenable to distributed computing. We illustrate that popular stationary kernels can be well approximated within our framework, making *string GPs* a scalable alternative to commonly used GP models. Finally, we derive the joint law of a *string GP* and its gradient, thereby allowing for explanatory analysis on the learned latent function.

The rest of the chapter is structured as follows. In section 4.2 we review recent advances on Gaussian processes in relation to inference on large datasets. In section 4.3 we formally construct *string GPs* and derive some important results. In section 4.4 we

provide detailed illustrative and theoretical comparisons between *string GPs* and the *standard GP paradigm*. We defer the detailed discussion of inference techniques under *string GP* priors to chapter 5, where we propose methods for automatically uncovering local patterns from datasets with time complexity and memory requirement that are linear in the size of the dataset.

4.2 Related Work

The two primary drawbacks of the *standard GP paradigm* on large scale problems are the lack of scalability resulting from postulating a full multivariate Gaussian prior on function values at *all* training inputs, and the difficulty postulating *a priori* a class of covariance functions capable of capturing intricate and often local patterns likely to occur in large datasets. A tremendous amount of work has been published that attempt to address either of the aforementioned limitations. However, scalability is often achieved either through approximations or for specific applications, and nonstationarity is usually introduced at the expense of scalability, again for specific applications.

Scalability Through Structured Approximations

As far as scalability is concerned, sparse GP methods have been developed that approximate the multivariate Gaussian probability density function (pdf) over training data with the marginal over a smaller set of inducing points multiplied by an approximate conditional pdf (Lawrence et al. (2003); Seeger (2003a,b); Smola and Bartlett (2001); Snelson and Ghahramani (2006)). This approximation yields a time complexity linear—rather than cubic—in the data size and squared in the number of inducing points. We refer to Quinonero-Candela and Rasmussen (2005) for a review of sparse GP approximations. More recently, Hensman et al. (2013, 2015) combined sparse GP methods with Stochastic Variational Inference (Hoffman et al. (2013)) for GP regression and GP classification. However, none of these sparse GP methods addresses the selection of the number of inducing points (and the size of the minibatch in the case of Hensman et al. (2013, 2015)), although this may greatly affect scalability. More importantly, although these methods do not impose strong restrictions on the covariance function of the GP model to approximate, they do not address the need for flexible covariance functions inherent to large scale problems, which are more likely to exhibit intricate and local patterns, and applications considered by the authors typically use the vanilla Gaussian kernel.

Lazaro-Gredilla et al. (2010) proposed approximating stationary kernels with truncated Fourier series in Gaussian process regression. An interpretation of the resulting sparse spectrum Gaussian process model as Bayesian basis function regression with a finite number K of trigonometric basis functions allows making inference in time complexity and memory requirement that are both linear in the size of the training sample. However, this model has two major drawbacks. Firstly, it is prone to overfitting. In effect, the learning machine will aim at inferring the K major spectral frequencies evidenced in the training data. This will only lead to appropriate prediction out-of-sample when the underlying latent phenomenon can be appropriately characterised by a finite discrete spectral decomposition that is expected to be the same everywhere on the domain. Secondly, this model implicitly postulates that the covariance between the values of the GP at two points does not vanish as the distance between the points becomes arbitrarily large. This imposes *a priori* the view that the underlying function is highly structured, which might be unrealistic in many real-life non-periodic applications. This approach is generalised by the so-called *random Fourier features* methods (Le et al. (2013); Rahimi and Recht (2007); Yang et al. (2015)). Unfortunately all existing random Fourier features methods give rise to stationary covariance functions, which might not be appropriate for datasets exhibiting local patterns.

The bottleneck of inference in the *standard GP paradigm* remains inverting and computing the determinant of a covariance matrix, normally achieved through the Cholesky decomposition or Singular Value Decomposition. Methods have been developed that speed-up these decompositions through low rank approximations (Williams and Seeger (2001)) or by exploiting specific structures in the covariance function and in the input data (Saatchi (2011); Wilson et al. (2014)), which typically give rise to Kronecker or Toeplitz covariance matrices. While the Kronecker method used by Saatchi (2011) and Wilson et al. (2014) is restricted to inputs that form a Cartesian grid and to separable kernels[2], low rank approximations such as the Nyström method used by Williams and Seeger (2001) modify the covariance function and hence the functional prior in a non-trivial way. Methods have also been proposed to interpolate the covariance matrix on a uniform or Cartesian grid in order to benefit from some of the computational gains of Toeplitz and Kronecker techniques even when the input space is not structured (Wilson and Nickisch (2015)). However, none of these solutions is general as they require that either the covariance function be separable (Kronecker techniques), or the

[2]That is multivariate kernel that can be written as product of univariate kernels.

covariance function be stationary and the input space be one-dimensional (Toeplitz techniques).

Scalability Through Data Distribution

A family of methods have been proposed to scale-up inference in GP models that are based on the observation that it is more computationally efficient to compute the pdf of K independent small Gaussian vectors with size n than to compute the pdf of a single bigger Gaussian vector of size nK. For instance, Kim et al. (2005) and Gramacy and Lee (2008) partitioned the input space, and put independent stationary GP priors on the restrictions of the latent function to the subdomains forming the partition, which can be regarded as independent *local GP experts*. Kim et al. (2005) partitioned the domain using Voronoi tessellations, while Gramacy and Lee (2008) used tree based partitioning. These two approaches are provably equivalent to postulating a (nonstationary) GP prior on the whole domain that is discontinuous along the boundaries of the partition, which might not be desirable if the latent function we would like to infer is continuous, and might affect predictive accuracy. The more local experts there are, the more scalable the model will be but the more discontinuities the latent function will have, and subsequently the less accurate the approach will be.

Mixtures of Gaussian process experts models (MoE) (Meeds and Osindero (2006); Rasmussen and Ghahramani (2001); Ross and Dy (2013); Tresp (2001)) provide another implementation of this idea. MoE models assume that there are multiple latent functions to be inferred from the data, on which it is placed independent GP priors, and each training input is associated to one latent function. The number of latent functions and the repartition of data between latent functions can then be performed in a full Bayesian nonparametric fashion (Rasmussen and Ghahramani (2001); Ross and Dy (2013)). When there is a single continuous latent function to be inferred, as it is the case for most regression models, the foregoing Bayesian nonparametric approach will learn a single latent function, thereby leading to a time complexity and a memory requirement that are the same as in the *standard GP paradigm*, which defies the scalability argument.

The last implementation of the idea in this section consists of distributing the training data over multiple independent but identical GP models. In regression problems, examples include the *Bayesian Committee Machines* (BCM) of Tresp (2000), the *generalized product of experts* (gPoE) model of Cao and Fleet (2014), and the *robust Bayesian Committee Machines* (rBCM) of Deisenroth and Ng (2015). These models propose splitting the training data in small subsets, each subset being assigned to a different

GP regression model—referred to as an expert—that has the same hyper-parameters as the other experts, although experts are assumed to be mutually independent. Training is performed by maximum marginal likelihood, with time complexity (resp. memory requirement) linear in the number of experts and cubic (resp. squared) in the size of the largest dataset processed by an expert. Predictions are then obtained by aggregating the predictions of all GP experts in a manner that is specific to the method used (that is the BCM, the gPoE or the rBCM). However, these methods present major drawbacks in the training and testing procedures. In effect, the assumption that experts have identical hyper-parameters is inappropriate for datasets exhibiting local patterns. Even if one would allow GP experts to be driven by different hyper-parameters, learned hyper-parameters would lead to overly simplistic GP experts and poor aggregated predictions when the number of training inputs assigned to each expert is small—this is a direct consequence of the (desirable) fact that maximum marginal likelihood GP regression abides by Occam's razor. Another critical pitfall of BCM, gPoE and rBCM is that their methods for aggregating expert predictions are Kolmogorov *inconsistent*. For instance, denoting $\hat{p}$ the predictive distribution in the BCM, and it can be easily seen from Equations (2.4) and (2.5) in Tresp (2000) that the predictive distribution $\hat{p}(f(x_1^*)|\mathcal{D})$ (resp. $\hat{p}(f(x_2^*)|\mathcal{D})$)[3] provided by the aggregation procedure of the BCM is *not* the marginal over $f(x_2^*)$ (resp. over $f(x_1^*)$) of the multivariate predictive distribution $\hat{p}(f(x_1^*), f(x_2^*)|\mathcal{D})$ obtained from experts multivariate predictions $p_k(f(x_1^*), f(x_2^*)|\mathcal{D})$ using the same aggregation procedure: $\hat{p}(f(x_1^*)|\mathcal{D}) \neq \int \hat{p}(f(x_1^*), f(x_2^*)|\mathcal{D}) df(x_2^*)$. Without Kolmogorov consistency, it is impossible to make principled Bayesian inference of latent function values. A principled Bayesian nonparametric model should not provide predictions about $f(x_1^*)$ that differ depending on whether or not one is also interested in predicting other values $f(x_i^*)$ simultaneously. This pitfall might be the reason why Cao and Fleet (2014) and Deisenroth and Ng (2015) restricted their expositions to predictive distributions about a single function value $p(f(x^*)|\mathcal{D})$, although their procedures (Equation (4) in Cao and Fleet (2014) and Equation (20) in Deisenroth and Ng (2015)) are easily extended to posterior distributions over multiple function values. These extensions would also be Kolmogorov *inconsistent*, and restricting the predictions to be of exactly one function value is unsatisfactory as it does not allow determining the posterior covariance between function values at two test inputs.

[3]Here f is the latent function to be inferred, x_1^*, x_2^* are test points and $\mathcal{D}$ denotes training data.

Expressive Stationary Kernels

In regards to flexibly handling complex patterns likely to occur in large datasets, Wilson and Adams (2013) introduced a class of expressive stationary kernels obtained by summing up convolutions of Gaussian basis functions with Dirac delta functions in the spectral domain. The sparse spectrum kernel can be thought of as the special case where the convolving Gaussian is degenerate. Although such kernels perform particularly well in the presence of globally repeated patterns in the data, their stationarity limits their utility on datasets with local patterns. Moreover the proposed covariance functions generate infinitely differentiable random functions, which might be too restrictive in some applications.

Application-Specific Nonstationary Kernels

As for nonstationary kernels, Paciorek and Schervish (2004) proposed a method for constructing nonstationary covariance functions from any stationary one that involves introducing n input dependent $d \times d$ covariance matrices that will be inferred from the data. Plagemann et al. (2008) proposed a faster approximation to the model of Paciorek and Schervish (2004). However, both approaches scale poorly with the input dimension and the data size as they have time complexity $\mathcal{O}\left(\max(nd^3, n^3)\right)$. Schmidt and O'Hagan (2003) introduced kernels that can be regarded as stationary after a non-linear transformation d on the input space: $k(x, x') = h\left(\|d(x) - d(x')\|\right)$, where h is positive semi-definite. Although for a given deterministic function d the kernel k is nonstationary, Schmidt and O'Hagan (2003) put a GP prior on d with mean function $m(x) = x$ and covariance function invariant under translation, which unfortunately leads to a kernel that is (unconditionally) stationary, albeit more flexible than $h\left(\|x - x'\|\right)$. To model nonstationarity, Adams and Stegle (2008) introduced a functional prior of the form $y(x) = f(x) \exp g(x)$ where f is a stationary GP and g is some scaling function on the domain. For a given non-constant function g such a prior indeed yields a nonstationary Gaussian process. However, when a stationary GP prior is put on the function g as Adams and Stegle (2008) did, the resulting functional prior $y(x) = f(x) \exp g(x)$ becomes stationary. The piecewise GP (Kim et al. (2005)) and treed GP (Gramacy and Lee (2008)) models previously discussed also introduce nonstationarity. The authors' premise is that heterogeneous patterns might be locally homogeneous. However, as previously discussed such models are inappropriate for modelling continuous latent functions.

Our Approach

The approach we propose in this chapter for inferring latent functions in large scale problems, exhibiting locally homogeneous patterns, consists of constructing a novel class of *smooth*, *nonstationary* and *flexible* stochastic processes we refer to as *string Gaussian processes* (*string GPs*), whose finite dimensional marginals are structured enough so that full Bayesian nonparametric inference scales linearly with the sample size, without resorting to approximations. Our approach is analogous to MoE models in that, when the input space is one-dimensional, a *string GP* can be regarded as a *collaboration of local GP experts* on non-overlapping supports, that implicitly exchange soft messages with one another, and that are independent conditional on the aforementioned messages. Each local GP expert only shares just enough information with adjacent local GP experts for the whole stochastic process to be sufficiently smooth (for instance continuously differentiable), which is an important improvement over MoE models as the latter generate discontinuous latent functions. These messages will take the form of boundary conditions, conditional on which each local GP expert will be independent from any other local GP expert. Crucially, unlike the BCM, the gPoE and the rBCM, we do not assume that local GP experts share the same prior structure (that is mean function, covariance function, or hyper-parameters). This allows each local GP expert to flexibly learn local patterns from the data if there are any, while preserving global smoothness, which will result in improved accuracy. Similarly to MoEs, the computational gain in our approach stems from the fact that the conditional independence of the local GP experts conditional on shared boundaries conditions will enable us to write the joint distribution over function and derivative values at a large number of inputs as the product of pdfs of much smaller Gaussian vectors. The resulting effect on time complexity is a decrease from $\mathcal{O}(n^3)$ to $\mathcal{O}(\max_k n_k^3)$, where $n = \sum_k n_k,\ n_k \ll n$. In fact, in Chapter 5 we will propose Reversible Jump Monte Carlo Markov Chain (RJ-MCMC) inference methods that achieve memory requirement and time complexity $\mathcal{O}(n)$, without any loss of flexibility. All these results are preserved by our extension of *string GPs* to multivariate input spaces, which we will occasionally refer to as *membrane Gaussian processes* (or membrane GPs). Unlike the BCM, the gPoE and the rBCM, the approach we propose in this chapter is Kolmogorov consistent, and enables principled inference of the posterior distribution over the values of the latent function at multiple test inputs. We will refer to our approach as the *string GP paradigm*.

4.3 The Model

In this section we formally construct *string* Gaussian processes, and we provide some important theoretical results including smoothness, and the joint law of *string GPs* and their gradients. We construct *string GPs* indexed on $\mathbb{R}$, before generalising to *string GPs* indexed on $\mathbb{R}^d$, which we will occasionally refer to as *membrane GPs*. We start by considering the joint law of a differentiable GP on an interval and its derivative, and introducing some related notions that we will use in the construction of *string GPs*.

Proposition 4.1 ***(Derivative Gaussian processes)***
Let I be an interval, $k : I \times I \to \mathbb{R}$ a $\mathcal{C}^2$ symmetric positive semi-definite function[4]*, $m : I \to \mathbb{R}$ a $\mathcal{C}^1$ function.*
(A) There exists a $\mathbb{R}^2$-valued stochastic process $(D_t)_{t\in I}$, $D_t = (z_t, z'_t)$, such that for all $t_1, \dots, t_n \in I$,

$$(z_{t_1}, \dots, z_{t_n}, z'_{t_1}, \dots, z'_{t_n})$$

is a Gaussian vector with mean

$$\left(m(t_1), \dots, m(t_n), \frac{dm}{dt}(t_1), \dots, \frac{dm}{dt}(t_n)\right)$$

and covariance matrix such that

$$cov(z_{t_i}, z_{t_j}) = k(t_i, t_j), \quad cov(z_{t_i}, z'_{t_j}) = \frac{\partial k}{\partial y}(t_i, t_j), \quad \text{and} \quad cov(z'_{t_i}, z'_{t_j}) = \frac{\partial^2 k}{\partial x \partial y}(t_i, t_j).$$

We herein refer to $(D_t)_{t\in I}$ as a ***derivative Gaussian process****.*
(B) $(z_t)_{t\in I}$ is a Gaussian process with mean function m, covariance function k and that is $\mathcal{C}^1$ in the L^2 (mean square) sense.
(C) $(z'_t)_{t\in I}$ is a Gaussian process with mean function $\frac{dm}{dt}$ and covariance function $\frac{\partial^2 k}{\partial x \partial y}$. Moreover, $(z'_t)_{t\in I}$ is the L^2 derivative of the process $(z_t)_{t\in I}$.

Proof See Appendix C.1. ■

We will say of a kernel k that it is **degenerate at** a when a *derivative Gaussian process* $(z_t, z'_t)_{t\in I}$ with kernel k is such that z_a and z'_a are perfectly correlated[5], that is

$$|\text{corr}(z_a, z'_a)| = 1.$$

[4] $\mathcal{C}^1$ (resp. $\mathcal{C}^2$) functions denote functions that are once (resp. twice) continuously differentiable on their domains.

[5] Or equivalently when the Gaussian vector (z_a, z'_a) is degenerate.

As an example, the linear kernel $k(u,v) = \sigma^2(u-c)(v-c)$ is degenerate at 0. Moreover, we will say of a kernel k that it is **degenerate at** b **given** a when it is not degenerate at a and when the *derivative Gaussian process* $(z_t, z'_t)_{t\in I}$ with kernel k is such that the variances of z_b and z'_b conditional on (z_a, z'_a) are both zero[6]. For instance, the periodic kernel proposed by MacKay (1998) with period T is degenerate at $u+T$ given u.

An important subclass of *derivative Gaussian processes* in our construction are the processes resulting from conditioning paths of a *derivative Gaussian process* to take specific values at certain times $(t_1, \dots, t_c)$. We herein refer to those processes as ***conditional derivative Gaussian process***. As an illustration, when k is $\mathcal{C}^3$ on $I \times I$ with $I = [a,b]$, and neither degenerate at a nor degenerate at b given a, the *conditional derivative Gaussian process* on $I = [a,b]$ with unconditional mean function m and unconditional covariance function k that is conditioned to start at $(\tilde{z}_a, \tilde{z}'_a)$ is the *derivative Gaussian process* with mean function

$$\forall\, t \in I, \quad m_c^a(t; \tilde{z}_a, \tilde{z}'_a) = m(t) + \tilde{\mathbf{K}}_{t;a}\mathbf{K}_{a;a}^{-1}\begin{bmatrix} \tilde{z}_a - m(a) \\ \tilde{z}'_a - \frac{dm}{dt}(a) \end{bmatrix}, \tag{4.1}$$

and covariance function k_c^a that reads

$$\forall\, t, s \in I, \quad k_c^a(t,s) = k(t,s) - \tilde{\mathbf{K}}_{t;a}\mathbf{K}_{a;a}^{-1}\tilde{\mathbf{K}}_{s;a}^T \tag{4.2}$$

where $\mathbf{K}_{u;v} = \begin{bmatrix} k(u,v) & \frac{\partial k}{\partial y}(u,v) \\ \frac{\partial k}{\partial x}(u,v) & \frac{\partial^2 k}{\partial x \partial y}(u,v) \end{bmatrix}$, and $\tilde{\mathbf{K}}_{t;a} = \begin{bmatrix} k(t,a) & \frac{\partial k}{\partial y}(t,a) \end{bmatrix}$. Similarly, when the process is conditioned to start at $(\tilde{z}_a, \tilde{z}'_a)$ and to end at $(\tilde{z}_b, \tilde{z}'_b)$, the mean function reads

$$\forall\, t \in I, \quad m_c^{a,b}(t; \tilde{z}_a, \tilde{z}'_a, \tilde{z}_b, \tilde{z}'_b) = m(t) + \tilde{\mathbf{K}}_{t;(a,b)}\mathbf{K}_{(a,b);(a,b)}^{-1}\begin{bmatrix} \tilde{z}_a - m(a) \\ \tilde{z}'_a - \frac{dm}{dt}(a) \\ \tilde{z}_b - m(b) \\ \tilde{z}'_b - \frac{dm}{dt}(b) \end{bmatrix}, \tag{4.3}$$

and the covariance function $k_c^{a,b}$ reads

$$\forall\, t, s \in I, \quad k_c^{a,b}(t,s) = k(t,s) - \tilde{\mathbf{K}}_{t;(a,b)}\mathbf{K}_{(a,b);(a,b)}^{-1}\tilde{\mathbf{K}}_{s;(a,b)}^T, \tag{4.4}$$

[6]Or equivalently when the Gaussian vector (z_a, z'_a) is not degenerate but (z_a, z'_a, z_b, z'_b) is.

where $\mathbf{K}_{(a,b);(a,b)} = \begin{bmatrix} \mathbf{K}_{a;a} & \mathbf{K}_{a;b} \\ \mathbf{K}_{b;a} & \mathbf{K}_{b;b} \end{bmatrix}$, and $\tilde{\mathbf{K}}_{t;(a,b)} = \begin{bmatrix} \tilde{\mathbf{K}}_{t;a} & \tilde{\mathbf{K}}_{t;b} \end{bmatrix}$. It is important to note that both $\mathbf{K}_{a;a}$ and $\mathbf{K}_{(a,b);(a,b)}$ are indeed invertible because the kernel is assumed to be neither degenerate at a nor degenerate at b given a. Hence, the support of (z_a, z'_a, z_b, z'_b) is $\mathbb{R}^4$, and any function and derivative values can be used for conditioning. Figure 4.1 illustrates example independent draws from a *conditional derivative Gaussian process*.

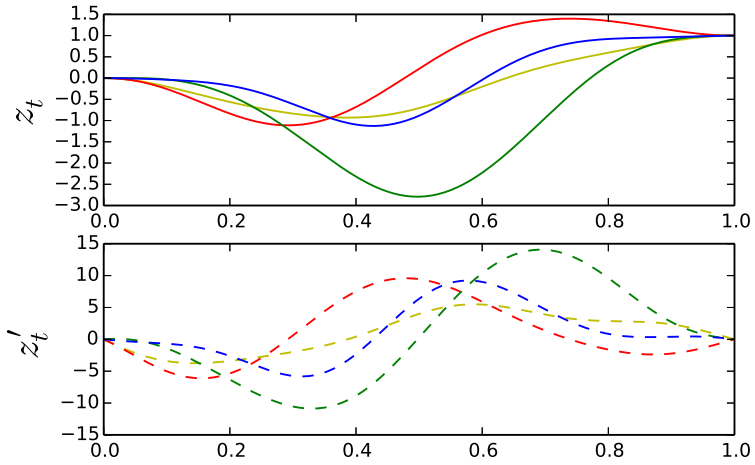

Fig. 4.1 Draws from a conditional derivative GP conditioned to start at 0 with derivative 0 and to finish at 1.0 with derivative 0.0. The unconditional kernel is the squared exponential kernel with variance 1.0 and input scale 0.2.

4.3.1 String Gaussian Processes on $\mathbb{R}$

The intuition behind string Gaussian processes on an interval is built via collaborative local GP experts we refer to as *strings* that are connected but independent of each other conditional on some regularity conditions at the boundaries. While each string is tasked with representing local patterns in the data, a string only shares the *states* of its extremities (value and derivative) with adjacent strings. Our aim is to preserve global smoothness and limit the amount of information shared between strings, thus reducing computational complexity. Furthermore, the conditional independence between strings will allow for distributed inference, greater flexibility and principled nonstationarity construction.

The following theorem at the core of our framework establishes that it is possible to connect together GPs on a partition of an interval I, in a manner consistent enough that the newly constructed stochastic object will be a stochastic process on I and in a manner restrictive enough that any two connected GPs will share just enough information to ensure that the constructed stochastic process is continuously differentiable ($\mathcal{C}^1$) on I in the L^2 sense.

Theorem 4.2 ***(String Gaussian process)***
Let $a_0 < \cdots < a_k < \cdots < a_K$, $I = [a_0, a_K]$ and let $\mathcal{N}(x|\mu, \Sigma)$ be the multivariate Gaussian density with mean vector μ and covariance matrix Σ. Furthermore, let $(m_k : [a_{k-1}, a_k] \to \mathbb{R})_{k\in[1..K]}$ be $\mathcal{C}^1$ functions, and $(k_k : [a_{k-1}, a_k] \times [a_{k-1}, a_k] \to \mathbb{R})_{k\in[1..K]}$ be $\mathcal{C}^3$ symmetric positive semi-definite functions, neither degenerate at a_{k-1}, nor degenerate at a_k given a_{k-1}.
(A) There exists an $\mathbb{R}^2$-valued stochastic process $(SD_t)_{t\in I}$, $SD_t = (z_t, z'_t)$ satisfying the following conditions:
1) The probability density of $(SD_{a_0}, \ldots, SD_{a_K})$ reads:

$$p_b(x_0, \ldots, x_K) := \prod_{k=0}^{K} \mathcal{N}\left(x_k|\mu_k^b, \Sigma_k^b\right) \tag{4.5}$$

$$\text{where:} \quad \Sigma_0^b = {}_1\boldsymbol{K}_{a_0;a_0}, \quad \forall\, k > 0 \quad \Sigma_k^b = {}_k\boldsymbol{K}_{a_k;a_k} - {}_k\boldsymbol{K}_{a_k;a_{k-1}}\ {}_k\boldsymbol{K}^{-1}_{a_{k-1};a_{k-1}}\ {}_k\boldsymbol{K}^{T}_{a_k;a_{k-1}}, \tag{4.6}$$

$$\mu_0^b = {}_1\boldsymbol{M}_{a_0}, \quad \forall\, k > 0 \quad \mu_k^b = {}_k\boldsymbol{M}_{a_k} + {}_k\boldsymbol{K}_{a_k;a_{k-1}}\ {}_k\boldsymbol{K}^{-1}_{a_{k-1};a_{k-1}}(x_{k-1} - {}_k\boldsymbol{M}_{a_{k-1}}), \tag{4.7}$$

$$\text{with} \quad {}_k\boldsymbol{K}_{u;v} = \begin{bmatrix} k_k(u,v) & \frac{\partial k_k}{\partial y}(u,v) \\ \frac{\partial k_k}{\partial x}(u,v) & \frac{\partial^2 k_k}{\partial x \partial y}(u,v) \end{bmatrix}, \quad \text{and} \quad {}_k\boldsymbol{M}_u = \begin{bmatrix} m_k(u) \\ \frac{dm_k}{dt}(u) \end{bmatrix}.$$

*2) Conditional on $(SD_{a_k} = x_k)_{k\in[0..K]}$, the restrictions $(SD_t)_{t\in]a_{k-1},a_k[}$, $k \in [1..K]$ are **independent conditional derivative Gaussian processes**, respectively with unconditional mean function m_k and unconditional covariance function k_k and that are conditioned to take values x_{k-1} and x_k at a_{k-1} and a_k respectively. We refer to $(SD_t)_{t\in I}$ as a **string derivative Gaussian process**, and to its first coordinate $(z_t)_{t\in I}$ as a **string Gaussian process** namely,*

$$(z_t)_{t\in I} \sim \mathcal{SGP}(\{a_k\}, \{m_k\}, \{k_k\}).$$

*(B) The **string Gaussian process** $(z_t)_{t\in I}$ defined in (A) is $\mathcal{C}^1$ in the L^2 sense and its L^2 derivative is the process $(z'_t)_{t\in I}$ defined in (A).*

Proof See Appendix C.2. ■

In our *collaborative local GP experts* analogy, Theorem 4.2 stipulates that each local expert takes as message from the previous expert its left hand side boundary conditions, conditional on which it generates its right hand side boundary conditions, which it then passes on to the next expert. Conditional on their boundary conditions local experts are independent of each other, and resemble vibrating pieces of string on fixed extremities, hence the name *string Gaussian process*.

4.3.2 Pathwise Regularity

Thus far we have dealt with regularity only in the L^2 sense. However, we note that a sufficient condition for the process $(z'_t)_{t\in I}$ in Theorem 4.2 to be almost surely continuous (i.e. sample paths are continuous with probability 1) and to be the almost sure derivative of the string Gaussian process $(z_t)_{t\in I}$, is that the Gaussian processes on $I_k = [a_{k-1}, a_k]$ with mean and covariance functions $m_{ck}^{a_{k-1},a_k}$ and $k_{ck}^{a_{k-1},a_k}$ (as per Equations (4.3) and (4.4) with $m := m_k$ and $k := k_k$) are themselves almost surely $\mathcal{C}^1$ for every boundary condition[7]. We refer to (Adler and Taylor, 2011, Theorem 2.5.2) for a sufficient condition under which a $\mathcal{C}^1$ in L^2 Gaussian process is also almost surely $\mathcal{C}^1$. As the above question is provably equivalent to that of the almost sure continuity of a Gaussian process (see Adler and Taylor, 2011, p. 30), *Kolmogorov's continuity theorem* (see Øksendal, 2003, Theorem 2.2.3) provides a more intuitive, albeit stronger, sufficient condition than that of (Adler and Taylor, 2011, Theorem 2.5.2).

4.3.3 Simulation

Algorithm 4.1 illustrates sampling jointly from a string Gaussian process and its derivative on an interval $I = [a_0, a_K]$. We start off by sampling the string boundary conditions (z_{a_k}, z'_{a_k}) sequentially, conditional on which we sample the values of the stochastic process on each string. This we may do in parallel as the strings are independent of each other conditional on boundary conditions. The resulting time complexity is the sum of $\mathcal{O}(\max\ n_k^3)$ for sampling values within strings, and $\mathcal{O}(n)$ for sampling boundary conditions, where the sample size is $n = \sum_k n_k$. The memory requirement grows as the sum of $\mathcal{O}(\sum_k n_k^2)$, required to store conditional covariance matrices of the values within strings, and $\mathcal{O}(K)$ corresponding to the storage of covariance matrices of boundary conditions. In the special case where strings are all empty, that is inputs and boundary times are the same, the resulting time complexity

[7] The proof is provided in Appendix C.3.

and memory requirement are $\mathcal{O}(n)$. Figure 4.2 illustrates a sample from a string Gaussian process, drawn using this approach.

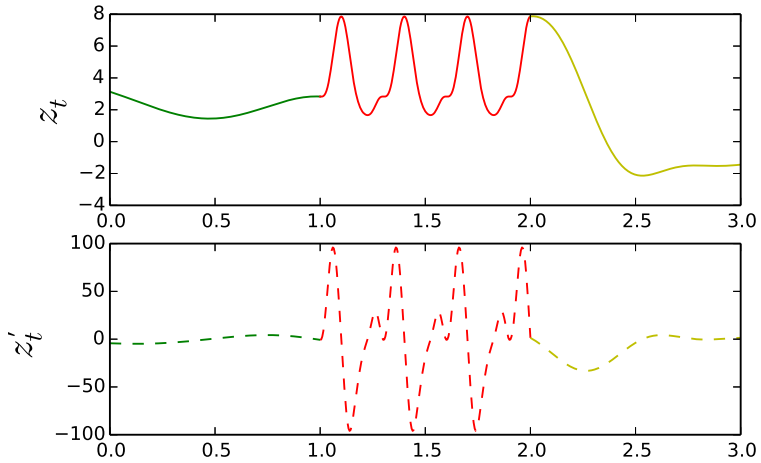

Fig. 4.2 Draw from a *string GP* (z_t) with 3 strings and its derivative (z'_t), under squared exponential kernels (green and yellow strings), and the periodic kernel of MacKay (1998) (red string).

4.3.4 String Gaussian Processes on $\mathbb{R}^d$

So far the input space has been assumed to be an interval. We generalise *string GPs* to hyper-rectangles in $\mathbb{R}^d$ as stochastic processes of the form:

$$f(t_1, \ldots, t_d) = \phi\left(z^1_{t_1}, \ldots, z^d_{t_d}\right), \tag{4.10}$$

where the *link function* $\phi : \mathbb{R}^d \to \mathbb{R}$ is a $\mathcal{C}^1$ function and (z^j_t) are d independent $(\perp)$ latent string Gaussian processes on intervals. We will occasionally refer to *string GPs* indexed on $\mathbb{R}^d$ with $d > 1$ as *membrane GPs* to avoid any ambiguity. We note that when $d = 1$ and when the link function is $\phi(x) = x$, we recover *string GPs* indexed on an interval as previously defined. When the *string GPs* (z^j_t) are a.s. $\mathcal{C}^1$, the *membrane GP* f in Equation (4.10) is also a.s. $\mathcal{C}^1$, and the partial derivative with respect to the j-th coordinate reads:

$$\frac{\partial f}{\partial t_j}(t_1, \ldots, t_d) = z^{j\prime}_{t_j} \frac{\partial \phi}{\partial t_j}\left(z^1_{t_1}, \ldots, z^d_{t_d}\right). \tag{4.11}$$

Algorithm 4.1 Simulation of a string derivative Gaussian process

Inputs: boundary times $a_0 < \cdots < a_K$, string times $\{t_j^k \in]a_{k-1}, a_k[\}_{j\in[1..n_k],k\in[1..K]}$, unconditional mean (resp. covariance) functions m_k (resp. k_k)
Output: $\{\ldots, z_{a_k}, z'_{a_k}, \ldots, z_{t_j^k}, z'_{t_j^k}, \ldots\}$.

Step 1: sample the boundary conditions sequentially.
for $k = 0$ **to** K **do**
Sample $(z_{a_k}, z'_{a_k}) \sim \mathcal{N}\left(\mu_k^b, \Sigma_k^b\right)$, with μ_k^b and Σ_k^b as per Equations (4.7) and (4.6).
end for
Step 2: sample the values on each string conditional on the boundary conditions in *parallel*.
parfor $k = 1$ **to** K **do**
Let ${}_k\mathbf{M}_u$ and ${}_k\mathbf{K}_{u;v}$ be as in Theorem 4.2,

$$ {}_k\Lambda = \begin{bmatrix} {}_k\mathbf{K}_{t_1^k;a_{k-1}} & {}_k\mathbf{K}_{t_1^k;a_k} \\ \ldots & \ldots \\ {}_k\mathbf{K}_{t_{n_k}^k;a_{k-1}} & {}_k\mathbf{K}_{t_{n_k}^k;a_k} \end{bmatrix} \begin{bmatrix} {}_k\mathbf{K}_{a_{k-1};a_{k-1}} & {}_k\mathbf{K}_{a_{k-1};a_k} \\ {}_k\mathbf{K}_{a_k;a_{k-1}} & {}_k\mathbf{K}_{a_k;a_k} \end{bmatrix}^{-1}, $$

$$ \mu_k^s = \begin{bmatrix} {}_k\mathbf{M}_{t_1^k} \\ \ldots \\ {}_k\mathbf{M}_{t_{n_k}^k} \end{bmatrix} + {}_k\Lambda \begin{bmatrix} z_{a_{k-1}} - m_k(a_{k-1}) \\ z'_{a_{k-1}} - \frac{dm_k}{dt}(a_{k-1}) \\ z_{a_k} - m_k(a_k) \\ z'_{a_k} - \frac{dm_k}{dt}(a_k) \end{bmatrix}, \tag{4.8} $$

$$ \Sigma_k^s = \begin{bmatrix} {}_k\mathbf{K}_{t_1^k;t_1^k} & \ldots & {}_k\mathbf{K}_{t_1^k;t_{n_k}^k} \\ \ldots & \ldots & \ldots \\ {}_k\mathbf{K}_{t_{n_k}^k;t_1^k} & \ldots & {}_k\mathbf{K}_{t_{n_k}^k;t_{n_k}^k} \end{bmatrix} - {}_k\Lambda \begin{bmatrix} {}_k\mathbf{K}_{t_1^k;a_{k-1}} & {}_k\mathbf{K}_{t_1^k;a_k} \\ \ldots & \ldots \\ {}_k\mathbf{K}_{t_{n_k}^k;a_{k-1}} & {}_k\mathbf{K}_{t_{n_k}^k;a_k} \end{bmatrix}^T. \tag{4.9} $$

Sample $(z_{t_1^k}, z'_{t_1^k} \ldots, z_{t_{n_k}^k}, z'_{t_{n_k}^k}) \sim \mathcal{N}(\mu_k^s, \Sigma_k^s)$.
end parfor

Thus in high dimensions, *string GPs* easily allow an explanation of the sensitivity of the learned latent function to inputs.

4.3.5 Choice of Link Function

Our extension of *string GPs* to $\mathbb{R}^d$ departs from the *standard GP paradigm* in that we did not postulate a covariance function on $\mathbb{R}^d \times \mathbb{R}^d$ directly. Doing so usually requires using a metric on $\mathbb{R}^d$, which is often problematic for heterogeneous input dimensions, as it introduces an arbitrary comparison between distances in each input dimension. This problem has been partially addressed by approaches such as Automatic Relevance Determination (ARD) kernels, that allow for a linear rescaling of input dimensions to

be learned jointly with kernel hyper-parameters. In the *string GP paradigm* however, inference under a *string GP* prior can be thought of as learning a coordinate system in which the latent function f resembles the link function ϕ, through non-linear rescaling of input dimensions. In particular, when ϕ is symmetric, the learned univariate *string GPs* (being interchangeable in ϕ) implicitly aim at normalising input data across dimensions, making *string GPs* naturally cope with heterogeneous datasets.

An important question arising from our extension is whether or not the link function ϕ needs to be learned to achieve a flexible functional prior. The flexibility of a *string GP* as a functional prior depends on both the *link function* and the covariance structures of the underlying *string GP* building blocks (z_t^j). To address the impact of the choice of ϕ on flexibility, we constrain the *string GP* building blocks by restricting them to be independent identically distributed *string GPs* with one string each (i.e. (z_t^j) are i.i.d Gaussian processes). Furthermore, we restrict ourselves to isotropic kernels as they provide a consistent basis for putting the same covariance structure in $\mathbb{R}$ and $\mathbb{R}^d$. One question we might then ask, for a given *link function* ϕ_0, is whether or not an isotropic GP indexed on $\mathbb{R}^d$ with covariance function k yields more flexible random surfaces than the stationary *string GP* $f(t_1, \dots, t_d) = \phi_0(z_{t_1}^1, \dots, z_{t_d}^d)$, where $(z_{t_j}^j)$ are stationary GPs indexed on $\mathbb{R}$ with the same covariance function k. If we find a *link function* ϕ_0 generating more flexible random surfaces than isotropic GP counterparts, that would suggest ϕ need not to be inferred in dimension $d > 1$ to be more flexible than any GP using one of the large number of commonly used isotropic kernels, among which squared exponential kernels, rational quadratic kernels, and Matérn kernels to name a few.

Before discussing whether such a ϕ_0 exists, we need to introduce a rigorous meaning to 'flexibility'. An intuitive qualitative definition of the flexibility of a stochastic process indexed on $\mathbb{R}^d$ is the ease with which it can generate surfaces with varying shapes from one random sample to another independent one. We recall that the tangent hyperplane to a $\mathcal{C}^1$ surface $y - f(x) = 0, x \in \mathbb{R}^d$ at some point $x_0 = (t_1^0, \dots, t_d^0)$ has equation $\nabla f(x_0)^T(x - x_0) - (y - f(x_0)) = 0$ and admits as normal vector $(\frac{\partial f}{\partial t_1}(t_1^0), \dots, \frac{\partial f}{\partial t_d}(t_d^0), -1)$. As tangent hyperplanes approximate a surface locally, a first criterion of flexibility for a random surface $y - f(x) = 0,\ x \in \mathbb{R}^d$ is the proclivity of the (random) direction of its tangent hyperplane at any point x — and hence the proclivity of $\nabla f(x)$ — to vary. This criterion alone however does not capture the difference between the local shapes of the random surface at two distinct points. A complementary second criterion of flexibility is the proclivity of the (random) directions of the tangent hyperplanes at any two distinct points $x_0, x_1 \in \mathbb{R}^d$ — and hence the proclivity of $\nabla f(x_0)$ and $\nabla f(x_1)$

— to be independent. The first criterion can be measured using the entropy of the gradient at a point, while the second criterion can be measured through the mutual information between the two gradients. The more flexible a stochastic process, the higher the entropy of its gradient at any point, and the lower the mutual information between its gradients at any two distinct points. This is formalised in the definition below.

Definition 4.3 ***(Flexibility of stochastic processes)***
Let f and g be two real valued, almost surely $\mathcal{C}^1$ stochastic processes indexed on $\mathbb{R}^d$, and whose gradients have a finite entropy everywhere (i.e. $\forall\ x,\ H(\nabla f(x)), H(\nabla g(x)) < \infty$). We say that f is more flexible than g if the following conditions are met:
1) $\forall\ x,\ H(\nabla f(x)) \geq H(\nabla g(x))$,
2) $\forall\ x \neq y,\ I(\nabla f(x); \nabla f(y)) \leq I(\nabla g(x); \nabla g(y))$,
where H is the entropy operator, and $I(X;Y) = H(X) + H(Y) - H(X,Y)$ stands for the mutual information between X and Y.

The following proposition establishes that the *link function* $\phi_s(x_1, \ldots, x_d) = \sum_{i=j}^{d} x_j$ yields more flexible stationary *string GPs* than their isotropic GP counterparts, thereby providing a theoretical underpinning for not inferring ϕ.

Proposition 4.4 ***(Additively separable* string GPs *are flexible)***
Let $k(x,y) := \rho\left(||x-y||^2_{L^2}\right)$ be a stationary covariance function generating a.s. $\mathcal{C}^1$ GP paths indexed on $\mathbb{R}^d$, $d > 0$, and ρ a function that is $\mathcal{C}^2$ on $]0, +\infty[$ and continuous at 0. Let $\phi_s(x_1, \ldots, x_d) = \sum_{j=1}^{d} x_j$, let $(z_t^j)_{t \in I^j,\ j \in [1..d]}$ be independent stationary Gaussian processes with mean 0 and covariance function k (where the L^2 norm is on $\mathbb{R}$), and let $f(t_1, \ldots, t_d) = \phi_s(z_{t_1}^1, \ldots, z_{t_d}^d)$ be the corresponding stationary string GP. Finally, let g be an isotropic Gaussian process indexed on $I^1 \times \cdots \times I^d$ with mean 0 and covariance function k (where the L^2 norm is on $\mathbb{R}^d$). Then:
1) $\forall\ x \in I^1 \times \cdots \times I^d,\ H(\nabla f(x)) = H(\nabla g(x))$,
2) $\forall\ x \neq y \in I^1 \times \cdots \times I^d,\ I(\nabla f(x); \nabla f(y)) \leq I(\nabla g(x); \nabla g(y))$.

Proof See Appendix C.4. ■

Although the link function need not be inferred in a full nonparametric fashion to yield comparable if not better results than most kernels used in the *standard GP paradigm*, for some problems certain link functions might outperform others. In section 4.4.2 we analyse a broad family of link functions, and argue that they extend successful anisotropic approaches such as the Automatic Relevance Determination (MacKay (1998)) and the additive kernels of Duvenaud et al. (2011). Moreover, in Chapter 5 we propose a scalable inference scheme applicable to any link function.

4.4 Comparison with the Standard GP Paradigm

We have already established that sampling *string GPs* scales better than sampling GPs under the *standard GP paradigm* and is amenable to distributed computing. We have also established that stationary additively separable *string GPs* are more flexible than their isotropic counterparts in the *standard GP paradigm*. In this section, we provide further theoretical results relating the *string GP paradigm* to the *standard GP paradigm*. Firstly we establish that *string GPs* with link function $\phi_s(x_1, \ldots, x_d) = \sum_{i=j}^{d} x_j$ are GPs. Secondly, we derive the global mean and covariance functions induced by the *string GP* construction for a variety of link functions. Thirdly, we provide a sense in which the *string GP paradigm* can be thought of as extending the *standard GP paradigm*. And finally, we show that the *string GP paradigm* may serve as a scalable approximation of commonly used stationary kernels.

4.4.1 Some String GPs are GPs

On one hand we note from Theorem 4.2 that the restriction of a *string GP* defined on an interval to the support of the first string — in other words the first local GP expert — is a Gaussian process. On the other hand, the messages passed on from one local GP expert to the next are not necessarily consistent with the unconditional law of the receiving local expert, so that overall a *string GP* defined on an interval, that is when looked at globally and unconditionally, might not be a Gaussian process. However, the following proposition establishes that some *string GPs* are indeed Gaussian processes.

Proposition 4.5 ***(Additively separable* string GPs *are GPs)***
String Gaussian processes on $\mathbb{R}$ *are Gaussian processes. Moreover, string Gaussian processes on* $\mathbb{R}^d$ *with link function* $\phi_s(x_1, \ldots, x_d) = \sum_{j=1}^{d} x_j$ *are also Gaussian processes.*

Proof The intuition behind this proof lies in the fact that if X is a multivariate Gaussian, and if conditional on X, Y is a multivariate Gaussian, providing that the conditional mean of Y depends linearly on X and the conditional covariance matrix of Y does not depend on X, the vector (X, Y) is jointly Gaussian. This will indeed be the case for our collaboration of local GP experts as the boundary conditions picked up by an expert from the previous will not influence the conditional covariance structure of the expert (the conditional covariance strucuture depends only on the partition of the domain, not the values of the boundary conditions) and will affect the mean linearly. See Appendix C.7 for the full proof. ■

The above result guarantees that commonly used closed form predictive equations under GP priors are still applicable under some *string GP* priors, providing the global mean and covariance functions, which we derive in the following section, are available. Proposition 4.5 also guarantees stability of the corresponding *string GPs* in the GP family under addition of independent Gaussian noise terms as in regression settings. Moreover, it follows from Proposition 4.5 that inference techniques developed for Gaussian processes can be readily used under *string GP* priors. In Chapter 5 we provide an additional MCMC scheme that exploits the conditional independence between strings to yield greater scalability and distributed inference.

4.4.2 String GP Kernels and String GP Mean Functions

The approach we have adopted in the construction of *string GPs* and *membrane GPs* did not require explicitly postulating a global mean function or covariance function. In Appendix C.8 we derive the global mean and covariance functions that result from our construction. The global covariance function could be used for instance as a stand-alone kernel in any kernel method, for instance GP models under the *standard GP paradigm*, which would provide a flexible and nonstationary alternative to commonly used kernels that may be used to learn local patterns in datasets—some successful example applications are provided in Chapter 5. That being said, adopting such a global approach should be limited to small scale problems as the conditional independence structure of *string GPs* does not easily translate into structures in covariance matrices over *string GP* values (without derivative information) that can be exploited to speed-up SVD or Cholesky decomposition. Crucially, marginalising out all derivative information in the distribution of *derivative string GP* values at some inputs would destroy any conditional independence structure, thereby limiting opportunities for scalable inference. In Chapter 5 we will provide a RJ-MCMC inference scheme that fully exploits the conditional independence structure in *string GPs* and scales to *very large* datasets.

4.4.3 Connection Between Multivariate String GP Kernels and Existing Approaches

We recall that for $n \leq d$, the n-th order *elementary symmetric polynomial* (Macdonald (1995)) is given by

$$e_0(x_1, \ldots, x_d) := 1, \quad \forall 1 \leq n \leq d \;\; e_n(x_1, \ldots, x_d) = \sum_{1 \leq j_1 < j_2 < \cdots < j_n \leq d} x_{j_1} \ldots x_{j_n}. \quad (4.12)$$

As an illustration,

$$e_1(x_1, \ldots, x_d) = \sum_{j=1}^{d} x_j = \phi_s(x_1, \ldots, x_d),$$

$$e_2(x_1, \ldots, x_d) = x_1x_2 + x_1x_3 + \cdots + x_1x_d + \cdots + x_{d-1}x_d,$$

$$\ldots$$

$$e_d(x_1, \ldots, x_d) = \prod_{j=1}^{d} x_j = \phi_p(x_1, \ldots, x_d).$$

Covariance kernels of *string GPs*, using as link functions *elementary symmetric polynomials* e_n, extend most popular approaches that combine unidimensional kernels over features for greater flexibility or cheaper design experiments.

The first order polynomial e_1 gives rise to additively separable Gaussian processes, that can be regarded as Bayesian nonparametric *generalised additive models* (GAM), particularly popular for their interpretability. Moreover, as noted by Durrande et al. (2012), additively separable Gaussian processes are considerably cheaper than alternate transformations in design experiments with high dimensional input spaces. In addition to the above, additively separable *string GPs* also allow postulating the existence of local properties in the experimental design process at no extra cost.

The d-th order polynomial e_d corresponds to a product of unidimensional kernels, also known as separable kernels. For instance, the popular squared exponential kernel is separable. Separable kernels have been successfully used on large scale inference problems where the inputs form a grid (Saatchi, 2011; Wilson et al., 2014), as they yield covariance matrices that are Kronecker products, leading to maximum likelihood inference in linear time complexity and with linear memory requirement. Separable kernels are often used in conjunction with the *automatic relevance determination* (ARD) model, to learn the relevance of features through global linear rescaling. However, ARD kernels might be limited in that we might want the relevance of a feature to depend on its value. As an illustration, the market value of a watch can be expected

to be a stronger indicator of its owner's wealth when it is in the top 1 percentile, than when it is in the bottom 1 percentile; the rationale being that possessing a luxurious watch is an indication that one can afford it, whereas possessing a cheap watch might be either an indication of lifestyle or an indication that one cannot afford a more expensive one. Separable *string GP* kernels extend ARD kernels, in that strings between input dimensions and within an input dimension may have unconditional kernels with different hyper-parameters, and possibly different functional forms, thereby allowing for *automatic local relevance determination* (ALRD).

More generally, using as link function the n-th order elementary symmetric polynomial e_n corresponds to the n-th order interaction of the *additive kernels* of Duvenaud et al. (2011). We also note that the class of link functions $\phi(x_1, \ldots, x_d) = \sum_{i=1}^{d} \sigma_i e_i(x_1, \ldots, x_d)$ yield full *additive kernels*. Duvenaud et al. (2011) noted that such kernels are 'exceptionally well-suited' to learn non-local structures in data. *String GPs* complement *additive kernels* by allowing them to learn local structures as well.

4.4.4 String GPs as Extension of the Standard GP Paradigm

The following proposition provides a perspective from which *string GPs* may be considered as extending Gaussian processes on an interval.

Proposition 4.6 ***(Extension of the standard GP paradigm)*** *Let $K \in \mathbb{N}^*$, let $I = [a_0, a_K]$ and $I_k = [a_{k-1}, a_k]$ be intervals with $a_0 < \cdots < a_K$. Furthermore, let $m : I \to \mathbb{R}$ be a $\mathcal{C}^1$ function, m_k the restriction of m to I_k, $h : I \times I \to \mathbb{R}$ a $\mathcal{C}^3$ symmetric positive semi-definite function, and h_k the restriction of h to $I_k \times I_k$. If*

$$(z_t)_{t \in I} \sim \mathcal{SGP}(\{a_k\}, \{m_k\}, \{h_k\}),$$

then

$$\forall\ k \in [1..K],\ \ (z_t)_{t \in I_k} \sim \mathcal{GP}(m, h).$$

Proof See Appendix C.5. ■

We refer to the case where unconditional string mean and kernel functions are restrictions of the same functions as in Proposition 4.6 as *uniform string GPs*. Although uniform *string GPs* are not guaranteed to be as much regular at boundary times as their counterparts in the *standard GP paradigm*, we would like to stress that they may well generate paths that are. In other words, the functional space induced by a uniform *string GP* on an interval extends the functional space of the GP with the

same mean and covariance functions m and h taken globally and unconditionally on the whole interval as in the *standard GP paradigm*. This allows for (but does not enforce) less regularity at the boundary times. When *string GPs* are used as functional prior, the posterior mean can in fact have more regularity at the boundary times than the continuous differentiability enforced in the *string GP paradigm*, providing such regularity is evidenced in the data.

We note from Proposition 4.6 that when m is constant and h is stationary, the restriction of the uniform *string GP* $(z_t)_{t\in I}$ to any interval whose interior does not contain a boundary time, the largest of which being the intervals $[a_{k-1}, a_k]$, is a stationary GP. We refer to such cases as *partition stationary string GPs*.

4.4.5 Commonly Used Covariance Functions and their String GP Counterparts

Considering the superior scalability of the *string GP paradigm*, which we may anticipate from the scalability of sampling *string GPs*, and which we will confirm in Chapter 5, a natural question that comes to mind is whether or not kernels commonly used in the *standard GP paradigm* can be well approximated by *string GP* kernels, so as to take advantage of the improved scalability of the *string GP paradigm*. We examine the distortions to commonly used covariance structures resulting from restricting strings to share only $\mathcal{C}^1$ boundary conditions, and from increasing the number of strings.

Figure 4.3 compares some popular stationary kernels on $[0,1]\times[0,1]$ (first column) to their uniform *string GP* kernel counterparts with 2, 4, 8 and 16 strings of equal length. The popular kernels considered are the squared exponential kernel (SE), the rational quadratic kernel $k_{RQ}(u,v) = \left(1 + \frac{2(u-v)^2}{\alpha}\right)^{-\alpha}$ with $\alpha = 1$ (RQ 1) and $\alpha = 5$ (RQ 5), the Matérn 3/2 kernel (MA 3/2), and the Matérn 5/2 kernel (MA 5/2), each with output scale (variance) 1 and input scale 0.5. Firstly, we observe that each of the popular kernels considered coincides with its uniform *string GP* counterparts regardless of the number of strings, so long as the arguments of the covariance function are less than an input scale apart. Except for the Matérn 3/2, the loss of information induced by restricting strings to share only $\mathcal{C}^1$ boundary conditions becomes noticeable when the arguments of the covariance function are more than 1.5 input scales apart, and the effect is amplified as the number of strings increases. As for the Matérn 3/2, no loss of information can been noticed, as further attests Table 4.1. In fact, this comes as no surprise given that we saw in Chapter 3 that stationary Matérn 3/2 GP are 1-Markov, that is the corresponding derivative Gaussian process is a Markov process so that the

vector (z_t, z_t') contains as much information as all *string GP* or derivative values prior to t. Table 4.1 provides some statistics on the absolute errors between each of the popular kernels considered and uniform *string GP* counterparts.

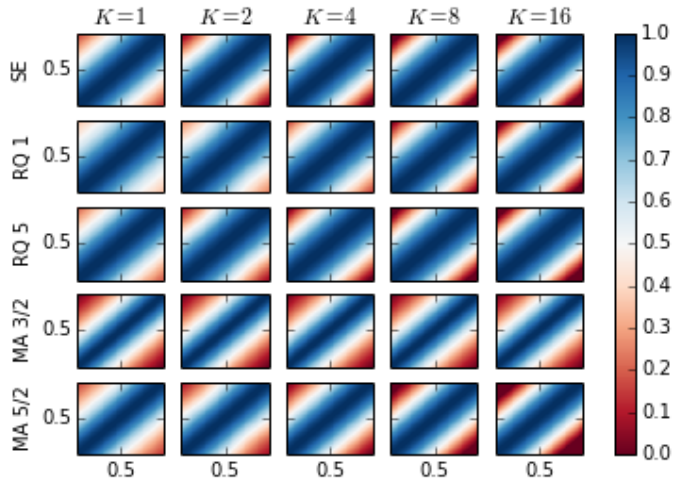

Fig. 4.3 Commonly used covariance functions on $[0, 1] \times [0, 1]$ with the same input and output scales (first column) and their uniform *string GP* counterparts with $K > 1$ strings of equal length.

	$K=2$			$K=4$			$K=8$			$K=16$		
	min	avg	max	min	avg	max	min	avg	max	min	avg	max
SE	0	0.01	0.13	0	0.02	0.25	0	0.03	0.37	0	0.04	0.44
RQ 1	0	0.01	0.09	0	0.03	0.20	0	0.05	0.37	0	0.07	0.52
RQ 5	0	0.01	0.12	0	0.02	0.24	0	0.04	0.37	0	0.05	0.47
MA 3/2	0	0	0	0	0	0	0	0	0	0	0	0
MA 5/2	0	0.01	0.07	0	0.03	0.15	0	0.05	0.29	0	0.08	0.48

Table 4.1 Minimum, average, and maximum absolute errors between some commonly used stationary covariance functions on $[0, 1] \times [0, 1]$ (with unit variance and input scale 0.5) and their uniform *string GP* counterparts with $K > 1$ strings of equal length.

4.5 Discussion

Some of the assumptions we have made in the construction of *string GPs* and *membrane GPs* can be relaxed, which we consider in detail below.

4.5.1 Stronger Global Regularity

We could have imposed more (multiple continuous differentiability) or less (continuity) regularity as boundary conditions in the construction of *string GPs*. We chose continuous differentiability as it is a relatively mild condition guaranteed by most popular kernels, and yet the corresponding treatment can be easily generalised to other regularity requirements. It is also possible to allow for discontinuity at a boundary time a_k by replacing μ_k^b and Σ_k^b in Equation (4.5) with ${}_kM_{a_k}$ and ${}_k\mathbf{K}_{a_k;a_k}$ respectively, or equivalently by preventing any communication between the k-th and the $k+1$-th strings. This would effectively be equivalent to having two independent *string GPs* on $[a_0, a_k]$ and $]a_k, a_K]$.

4.5.2 Differential Operators as Link Functions

Our framework can be further extended to allow differential operators as link functions, thereby considering the latent multivariate function to infer as the response of a differential system to independent univariate *string GP* excitations. The RJ-MCMC sampler we will propose in Chapter 5 would still work in this framework, with the only exception that, when the differential operator is of first order, the latent multivariate function will be continuous but not differentiable, except if global regularity is upgraded as discussed above. Moreover, Proposition 4.5 can be generalised to first order linear differential operators.

4.6 Summary

In this chapter we have introduced a novel class of smooth functional priors (or stochastic processes), which we refer to as *string GPs*, with the aim of simultaneously addressing the lack of scalability and the lack of flexibility of Bayesian kernel methods. Unlike existing approaches such as Gaussian process priors (Rasmussen and Williams (2005)) or Student-t process priors (Shah et al. (2014)), which are parameterised by *global* mean and covariance functions, and which postulate *fully dependent* finite-dimensional marginals, the alternative construction we propose adopts a *local* perspective and the

resulting finite-dimensional marginals exhibit *conditional independence* structures. Our local approach to constructing *string GPs* provides a principled way of postulating that the latent function we wish to learn might exhibit locally homogeneous patterns, while the conditional independence structures constitute the core ingredient needed for developing scalable inference methods. Moreover, we provide theoretical results relating our approach to Gaussian processes and some other popular kernel methods, and we illustrate that our approach can often be regarded as a more scalable and/or more flexible extension.

A few critical practical questions were purposely left out of the scope of this chapter, namely:

1. How can we go about using string GP functional priors to learn ***whether*** or not a dataset exhibits local patterns in the first place?

2. How can we go about using string GP functional priors to learn ***how many*** types of local patterns, if any, are evidenced in a dataset?

3. How can we go about using string GP functional priors to automatically detect ***where*** changes in local patterns occur, providing there are any?

4. How can we go about leveraging the conditional independence structures in string GPs to construct a Bayesian nonparametric ***inference scheme*** to answer the aforementioned questions, that ***scales to very large datasets***, and that is ***applicable to virtually any supervised function learning problem***?

Each of these questions will be answered in the next chapter, where we will also provide empirical evidence of the superiority of our approach over competing alternatives on a wide variety of machine learning tasks.

Chapter 5

Bayesian Inference under String Gaussian Process Priors

"Somewhere, something incredible is waiting to be known."

Carl Sagan

In this chapter we turn to discussing how *string GPs* may be used in some common machine learning problems. We start with maximum marginal likelihood inference for regression problems under additively separable *string GP* priors or GP priors driven by any *string GP* covariance function (as in section 4.4.2). We then derive a generic and scalable RJ-MCMC sampler for learning latent functions under any *string GP* functional prior, and any model likelihood that depends on the latent function which we want to infer solely through its values at a finite number of inputs. The RJ-MCMC sampler that we propose simultaneously addresses the learning of *whether* the dataset exhibits local patterns, *how many* types of local patterns are evidenced in the training dataset, and *where* do changes in patterns occur. Crucially, the overall time complexity and memory requirements of our sampler are both linear in the data size and the input dimension, so long as the evaluation of the model likelihood does not have time complexity or memory requirement that grows faster than linearly in the data size. This limit, on resources required to evaluate the model likelihood, is widely met in practice, for instance by all i.i.d. observation models, among which, reside regression models with additive white noise, logistic (or probit) regression, and multinomial logistic (or probit) regression to name but a few.

5.1 Maximum Marginal Likelihood for Small Scale Regression Problems

Firstly, we leverage the fact that additively separable *string GPs* are Gaussian processes to perform Bayesian nonparametric regression in the presence of local patterns in the data, using standard Gaussian process techniques (see Rasmussen and Williams, 2005, p.112 §5.4.1). We use as generative model

$$y_i = f(x_i) + \epsilon_i, \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma^2_{k_i}\right), \quad \sigma^2_{k_i} > 0, \quad x_i \in I^1 \times \cdots \times I^d, \quad y_i, \epsilon_i \in \mathbb{R},$$

we are given the training dataset $\mathcal{D} = \{\tilde{x}_i, \tilde{y}_i\}_{i\in[1..N]}$, and we place a mean-zero additively separable *string GP* prior on f, namely

$$f(x) = \sum_{j=1}^{d} z^j_{x[j]}, \quad (z^j_t) \sim \mathcal{SGP}\left(\{a^j_k\}, \{0\}, \{k^j_k\}\right), \quad \forall j < l,\ (z^j_t) \perp (z^l_t),$$

which we assume to be independent from the measurement noise process. Moreover, the noise terms are assumed to be independent, and the noise variance $\sigma^2_{k_i}$ affecting $f(x_i)$ is assumed to be the same for any two inputs whose coordinates lie on the same string intervals. Such a heteroskedastic noise model fits nicely within the *string GP paradigm*, can be very useful when the dimension of the input space is small, and may be replaced by the typical constant noise variance assumption in high-dimensional input spaces.

Let us define $\mathbf{y} = (\tilde{y}_1, \ldots, \tilde{y}_N)$, $\mathbf{X} = (\tilde{x}_1, \ldots, \tilde{x}_N)$, $\mathbf{f} = (f(\tilde{x}_1), \ldots, f(\tilde{x}_N))$ and let $\bar{\mathbf{K}}_{\mathbf{X};\mathbf{X}}$ denote the auto-covariance matrix of $\mathbf{f}$ (which we have derived in section 4.4.2), and let $\mathbf{D} = \mathrm{diag}(\{\sigma^2_{k_i}\})$ denote the diagonal matrix of noise variances. It follows that $\mathbf{y}$ is a Gaussian vector with mean 0 and auto-covariance matrix $\mathbf{K}_\mathbf{y} := \bar{\mathbf{K}}_{\mathbf{X};\mathbf{X}} + \mathbf{D}$ and that the log marginal likelihood reads:

$$\log p\left(\mathbf{y}\middle|\mathbf{X}, \{\sigma_{k_i}\}, \{\theta^j_k\}, \{a^j_k\}\right) = -\frac{1}{2}\mathbf{y}^T\mathbf{K}^{-1}_\mathbf{y}\mathbf{y} - \frac{1}{2}\log\det\left(\mathbf{K}_\mathbf{y}\right) - \frac{n}{2}\log 2\pi. \tag{5.1}$$

We obtain estimates of the string measurement noise standard deviations $\{\hat{\sigma}_{k_i}\}$, and estimates of the string hyper-parameters $\{\hat{\theta}^j_k\}$ by maximising the marginal likelihood for a given domain partition $\{a^j_k\}$, using gradient-based methods. We deduce the predictive mean and covariance matrix of the latent function values $\mathbf{f}^*$ at test points

$\mathbf{X}^*$, from the estimates $\{\hat{\theta}_k^j\}, \{\hat{\sigma}_{k_i}\}$ as

$$\mathrm{E}(\mathbf{f}^*|\mathbf{y}) = \bar{\mathbf{K}}_{\mathbf{X}^*;\mathbf{X}}\mathbf{K}_{\mathbf{y}}^{-1}\mathbf{y} \quad \text{and} \quad \mathrm{cov}(\mathbf{f}^*|\mathbf{y}) = \bar{\mathbf{K}}_{\mathbf{X}^*;\mathbf{X}^*} - \bar{\mathbf{K}}_{\mathbf{X}^*;\mathbf{X}}\mathbf{K}_{\mathbf{y}}^{-1}\bar{\mathbf{K}}_{\mathbf{X};\mathbf{X}^*},$$

using the fact that $(\mathbf{f}^*, \mathbf{y})$ is jointly Gaussian, and that the cross-covariance matrix between $\mathbf{f}^*$ and $\mathbf{y}$ is $\bar{\mathbf{K}}_{\mathbf{X}^*;\mathbf{X}}$ as the additive measurement noise is assumed to be independent from the latent process f.

Remarks

The above analysis and equations still hold when a GP prior is placed on f with one of the multivariate *string GP* kernels derived in section 4.4.2 as covariance function.

It is also worth noting from the derivation of *string GP* kernels in Appendix C.8 that the marginal likelihood Equation (5.1) is continuously differentiable in the locations of boundary times. Thus, for a given number of boundary times, the positions of the boundary times can be determined as part of the marginal likelihood maximisation. The derivatives of the marginal log-likelihood (Equation (5.1)) with respect to the aforementioned locations $\{a_k^j\}$ can be determined from the recursions of Appendix C.8, or approximated numerically by finite differences. The number of boundary times in each input dimension can then be learned by trading off model fit (the maximum marginal log likelihood) and model simplicity (the number of boundary times or model parameters), for instance using information criteria such as AIC and BIC. When the input dimension is large, it might be advantageous to further constrain the hypothesis space of boundary times before using information criteria, for instance by assuming that the number of boundary times is the same in each dimension. An alternative Bayesian nonparametric approach to learning the number of boundary times will be discussed in Section 5.4.

This method of inference cannot exploit the structure of *string GPs* to speed-up inference, and as a result it scales like the *standard GP paradigm*. In fact, any attempt to marginalize out univariate derivative processes, including in the prior, will inevitably destroy the conditional independence structure. Another perspective to this observation is found by noting from the derivation of global *string GP* covariance functions in Appendix C.8 that the conditional independence structure does not easily translate in a matrix structure that may be exploited to speed-up matrix inversion, and that marginalising out terms relating to derivatives processes as in Equation (5.1) can only make things worse.

5.2 Generic Reversible-Jump MCMC Sampler for Large Scale Inference

More generally, we consider learning a smooth real-valued latent function f, defined on a d-dimensional hyper-rectangle, under a generative model with likelihood $p(\mathcal{D}|\mathbf{f}, \mathbf{u})$, where $\mathbf{f}$ denotes values of f at training inputs points and $\mathbf{u}$ denotes other likelihood parameters that are not related to f. A large class of machine learning problems aiming at inferring a latent function have a likelihood model of this form. Examples include celebrated applications such as nonparametric regression and nonparametric binary classification problems, but also more recent applications such as learning a profitable portfolio generating-function in *stochastic portfolio theory* (Karatzas and Fernholz (2009)) from the data. In particular, we do not assume that $p(\mathcal{D}|\mathbf{f}, \mathbf{u})$ factorizes over training inputs. Extensions to likelihood models that depend on the values of multiple latent functions are straight-forward and will be discussed in section 5.3.

5.2.1 Prior Specification

We place a prior $p(\mathbf{u})$ on other likelihood parameters. For instance, in regression problems under a Gaussian noise model, $\mathbf{u}$ can be the noise variance and we may choose $p(\mathbf{u})$ to be the inverse-Gamma distribution for conjugacy. We place a mean-zero *string GP* prior on f

$$f(x) = \phi\left(z^1_{x[1]}, \ldots, z^d_{x[d]}\right), \qquad (z^j_t) \sim \mathcal{SGP}\left(\{a^j_k\}, \{0\}, \{k^j_k\}\right), \qquad \forall j < l,\ (z^j_t) \perp (z^l_t). \tag{5.2}$$

As discussed in section 4.3.5, the link function ϕ need not be inferred as the symmetric sum was found to yield a sufficiently flexible functional prior. Nonetheless, in this section we do not impose any restriction on the link function ϕ other than continuous differentiability. Denoting $\mathbf{z}$ the vector of univariate *string GP* processes and their derivatives, evaluated at all distinct input coordinate values, we may re-parameterise the likelihood as $p(\mathcal{D}|\mathbf{z}, \mathbf{u})$, with the understanding that $\mathbf{f}$ can be recovered from $\mathbf{z}$ through the link function ϕ. To complete our prior specification, we need to discuss the choice of boundary times $\{a^j_k\}$ and the choice of the corresponding unconditional kernel structures $\{k^j_k\}$. Before doing so, we would like to stress that key requirements of our sampler are that i) it should decouple the need for scalability from the need for flexibility, ii) it should scale linearly with the number of training and test inputs, and iii) the user should be able to express prior views on model complexity/flexibility in an intuitive way, but the sampler should be able to validate or invalidate the prior model

complexity from the data. While the motivations for the last two requirements are obvious, the first requirement is motivated by the fact that a massive dataset may well be more homogeneous than a much smaller dataset.

5.2.2 Scalable Choice of Boundary Times

To motivate our choice of boundary times that achieves great scalability, we first note that the evaluation of the likelihood, which will naturally be needed by the MCMC sampler, will typically have at least linear time complexity and linear memory requirement, as it will require performing computations that use each training sample at least once. Thus, the best we can hope to achieve overall is linear time complexity and linear memory requirement. Second, in MCMC schemes with functional priors, the time complexity and memory requirements for sampling from the posterior

$$p\left(\mathbf{f}|\mathcal{D}\right) \propto p\left(\mathcal{D}|\mathbf{f}\right) p(\mathbf{f})$$

are often the same as the resource requirements for sampling from the prior $p\left(\mathbf{f}\right)$, as evaluating the model likelihood is rarely the bottleneck. Finally, we note from Algorithm 4.1 that, when each input coordinate in each dimension is a boundary time, the sampling scheme has time complexity and memory requirement that are linear in the maximum number of unique input coordinates across dimensions, which is at most the number of training samples. In effect, each univariate derivative *string GP* is sampled in *parallel* at as many times as there are unique input coordinates in that dimension, before being combined through the link function. In a given input dimension, univariate *derivative string GP* values are sampled sequentially, one boundary time conditional on the previous. The foregoing sampling operation is very scalable not only asymptotically but also in absolute terms; it merely requires storing and inverting at most as many 2×2 matrices as the number of input points. We will evaluate the actual overall time complexity and memory requirement when we discuss our MCMC sampler in greater details. For now, we would like to stress that i) choosing each distinct input coordinate value as a boundary time in the corresponding input dimension before training is a perfectly valid choice, ii) we expect this choice to result in resource requirements that grow linearly with the sample size and iii) in the *string GP* theory we have developed thus far there is no requirement that two adjacent strings be driven by different kernel hyper-parameters.

5.2.3 Model Complexity Learning as a Change-Point Problem

The remark iii) above pertains to model complexity. In the simplest case, all strings are driven by the same kernel and hyper-parameters as it was the case in section 4.4.5, where we discussed how this setup departs from postulating the unconditional string covariance function k_k^j globally similarly to the *standard GP paradigm*. The more distinct unconditional covariance structures there are, the more complex the model is, as it may account for more types of local patterns. Thus, we may identify model complexity to the number of different kernel configurations across input dimensions. In order to learn model complexity, we require that some (but not necessarily all) strings share their kernel configuration.[1] Moreover, we require kernel membership to be dimension-specific in that two strings in different input dimensions may not explicitly share a kernel configuration in the prior specification, although the posterior distribution over their hyper-parameters might be similar if the data support it.

In each input dimension j, kernel membership is defined by a partition of the corresponding domain operated by a (possibly empty) set of change-points,[2] as illustrated in Figure 5.1. When there is no change-point as in Figure 5.1-(a), all strings are driven by the same kernel and hyper-parameters. Each change-point c_p^j induces a new kernel configuration θ_p^j that is shared by all strings whose boundary times a_k^j and a_{k+1}^j both lie in $[c_p^j, c_{p+1}^j[$. When one or multiple change-points c_p^j occur between two adjacent boundary times as illustrated in Figures 5.1-(b-d), for instance $a_k^j \leq c_p^j \leq a_{k+1}^j$, the kernel configuration of the string defined on $[a_k^j, a_{k+1}^j]$ is that of the largest change-point that lies in $[a_k^j, a_{k+1}^j]$ (see for instance Figure 5.1-(d)). For consistency, we denote θ_0^j the kernel configuration driving the first string in the j-th dimension; it also drives strings that come before the first change-point, and all strings when there is no change-point.

To place a prior on model complexity, it suffices to define a joint probability measure on the set of change-points and the corresponding kernel configurations. As kernel configurations are not shared across input dimensions, we choose these priors to be independent across input dimensions. Moreover, $\{c_p^j\}$ being a random collection of points on an interval whose number and positions are both random, it is *de facto* a point process (Daley and Vere-Jones (2008)). To keep the prior specification of change-points uninformative, it is desirable that conditional on the number of change-points, the positions of change-points be i.i.d. uniform on the domain. As for the number of

[1] That is, the functional form of the unconditional kernel k_k^j and its hyper-parameters.

[2] We would like to stress that change-points do not introduce new input points or boundary times, but solely define a partition of the domain of each input dimension.

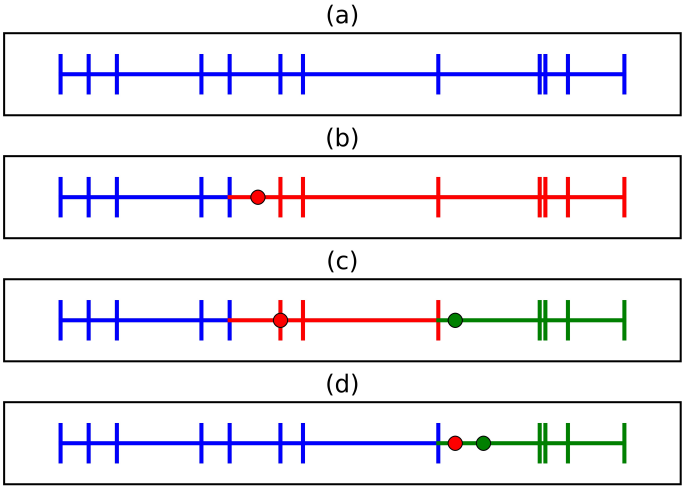

Fig. 5.1 Effects of domain partition through change-points (coloured circles), on kernel membership. Each vertical bar corresponds to a distinct boundary time a_k^j. For the same collection of boundary times, we consider four scenarios: (a) no partition, (b) partition of the domain in two by a single change-point that does not coincide with any existing boundary time, (c) partition of the domain in three by two change-points, one of which coincides with an existing boundary time, and (d) partition of the domain in two by two distinct change-points. In each scenario, kernel membership is illustrated by colour-coding. The colour of the interval between two consecutive boundary times a_k^j and a_{k+1}^j reflects what kernel configuration drives the corresponding string; in particular, the colour of the vertical bar corresponding to boundary time a_{k+1}^j determines what kernel configuration should be used to compute the conditional distribution of the value of the *derivative string GP* $(z_t^j, z_t^{j\prime})$ at a_{k+1}^j, given its value at a_k^j.

change-points, it is important that the support of its distribution not be bounded, so as to allow for an arbitrarily large model complexity if warranted. The two requirements above are satisfied by a homogeneous Poisson process or HPP (Daley and Vere-Jones (2008)) with constant intensity λ^j. More precisely, the prior probability measure on $\left(\{c_p^j, \theta_p^j\}, \lambda^j\right)$ is constructed as follows:

$$\begin{cases} \lambda^j \sim \Gamma(\alpha^j, \beta^j), \\ \{c_p^j\} \Big| \lambda^j \sim \text{HPP}(\lambda^j) \\ \theta_p^j[i] \Big| \{c_p^j\}, \lambda^j \overset{\text{i.i.d}}{\sim} \log \mathcal{N}(0, \rho^j) \\ \forall (j,p) \neq (l,q)\ \theta_p^j \perp \theta_q^l, \end{cases}, \tag{5.3}$$

where we choose the Gamma distribution Γ as prior on the intensity λ^j for conjugacy, we assume all kernel hyper-parameters are positive as is often the case in practice,[3] the coordinates of the hyper-parameters of a kernel configuration are assumed i.i.d., and kernel hyper-parameters are assumed independent between kernel configurations. Denoting the domain of the j-th input $[a^j, b^j]$, it follows from applying the laws of total expectation and total variance on Equation (5.3) that the expected number of change-points in the j-th dimension under our prior is

$$\mathrm{E}\left(\#\{c_p^j\}\right) = \left(b^j - a^j\right)\frac{\alpha^j}{\beta^j}, \tag{5.4}$$

and the variance of the number of change-points in the j-dimension under our prior is

$$\mathrm{Var}\left(\#\{c_p^j\}\right) = \left(b^j - a^j\right)\frac{\alpha^j}{\beta^j}\left(1 + \frac{(b^j - a^j)}{\beta^j}\right). \tag{5.5}$$

The two equations above may guide the user when setting the parameters α^j and β^j. For instance, these values may be set so that the expected number of change-points in a given input dimension be a fixed fraction of the number of boundary times in that input dimension, and so that the prior variance over the number of change-points be large enough that overall the prior isn't too informative.

We could have taken a different approach to construct our prior on change-points. In effect, assuming for the sake of the argument that the boundaries of the domain of the j-th input, namely a^j and b^j, are the first and last change-point in that input dimension, we note that the mapping

$$\left(\ldots, c_p^j, \ldots\right) \to \left(\ldots, p_p^j, \ldots\right) := \left(\ldots, \frac{c_{p+1}^j - c_p^j}{b^j - a^j}, \ldots\right)$$

defines a bijection between the set of possible change-points in the j-th dimension and the set of all discrete probability distributions. Thus, we could have placed as prior on $\left(\ldots, p_p^j, \ldots\right)$ a Dirichlet process (Ferguson (1973)), a Pitman-Yor process (Pitman and Yor (1997)), more generally *normalized completely random measures* (Kingman (1967)) or any other probability distribution over partitions. We prefer the point process approach primarily because it provides an easier way of expressing prior belief about model complexity through the expected number of change-points $\#\{c_p^j\}$, while remaining uninformative about positions thereof.

[3] This may easily be relaxed if needed, for instance by putting normal priors on parameters that may be negative and log-normal priors on positive parameters.

One might also be tempted to regard change-points in an input dimension j as inducing a partition, not of the domain $[a^j, b^j]$, but of the set of boundary times a_k^j in the same dimension, so that one may define a prior over kernel memberships through a prior over partitions of the set of boundary times. However, this approach would be inconsistent with the aim to learn local patterns in the data if the corresponding random measure is *exchangeable*. In effect, as boundary times are all input coordinates, local patterns may only arise in the data as a result of adjacent strings sharing kernel configurations. An exchangeable random measure would postulate a priori that two kernel membership assignments that have the same kernel configurations (i.e. the same number of configurations and the same set of hyper-parameters) and the same *number* of boundary times in each kernel cluster (although not exactly the same boundary times), are equally likely to occur, thereby possibly putting more probability mass on kernel membership assignments that do not respect boundary time adjacency. Unfortunately, *exchangeable* random measures (among which lie the Dirichlet process and the Pitman-Yor process) are by far more widely adopted by the machine learning community than non-exchangeable random measures. Thus, this approach might be perceived as overly complex. That being said, as noted by Foti and Williamson (2015), non-exchangeable normalized random measures may be regarded as Poisson point processes (with varying intensity functions) on some augmented spaces, which makes this choice of prior specification somewhat similar, but stronger (that is more informative) than the one we adopt in this chapter.

Before deriving the sampling algorithm, it is worth noting that the prior defined in Equation (5.3) does not admit a density with respect to the same base measure[4], as the number of change-points $\#\{c_p^j\}$, and subsequently the number of kernel configurations, may vary from one sample to another. Nevertheless, the joint distribution over the data $\mathcal{D}$ and all other model parameters is well defined and, as we will see later, we may leverage reversible-jump MCMC techniques (Green (1995); Green and Hastie (2009)) to construct a Markov chain that converges to the posterior distribution.

5.2.4 Overall Structure of the MCMC Sampler

To ease notations, we denote $\boldsymbol{c}$ the set of all change-points in all input dimensions, we denote $\boldsymbol{n} = \big(\dots, \#\{c_p^j\}, \dots\big) \in \mathbb{N}^d$ the vector of the numbers of change-points in each input dimension, we denote $\boldsymbol{\theta}$ the set of kernel hyper-parameters,[5] and $\boldsymbol{\rho} := (\dots, \rho^j, \dots)$

[4] By base measure we mean either the counting measure or Lebesgue's measure.

[5] To simplify the exposition, we assume without loss of generality that each kernel configuration has the same kernel functional form, so that configurations are defined by kernel hyper-parameters.

the vector of variances of the independent log-normal priors on $\boldsymbol{\theta}$. We denote $\boldsymbol{\lambda} := (\dots, \lambda^j, \dots)$ the vector of change-points intensities, we denote $\boldsymbol{\alpha} := (\dots, \alpha^j, \dots)$ and $\boldsymbol{\beta} := (\dots, \beta^j, \dots)$ the vectors of parameters of the Gamma priors we put on the change-points intensities across the d input dimensions, and we recall that $\mathbf{u}$ denotes the vector of likelihood parameters other than the values of the latent function f.

We would like to sample from the posterior distribution $p(\mathbf{f}, \mathbf{f}^*, \nabla\mathbf{f}, \nabla\mathbf{f}^* | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})$, where $\mathbf{f}$ and $\mathbf{f}^*$ are the vectors of values of the latent function f at training and test inputs respectively, and $\nabla\mathbf{f}, \nabla\mathbf{f}^*$ the corresponding gradients. Denoting $\mathbf{z}$ the vector of univariate *string GP* processes and their derivatives, evaluated at all distinct training and test input coordinate values, we note that to sample from $p(\mathbf{f}, \mathbf{f}^*, \nabla\mathbf{f}, \nabla\mathbf{f}^* | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})$, it suffices to sample from $p(\mathbf{z} | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})$, compute $\mathbf{f}$ and $\mathbf{f}^*$ using the link function, and compute the gradients using Equation (4.11). To sample from $p(\mathbf{z} | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})$, we may sample from the target distribution

$$\pi(\boldsymbol{n}, \boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{z}, \mathbf{u}) := p(\boldsymbol{n}, \boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{z}, \mathbf{u} | \mathcal{D}, \boldsymbol{\alpha}, \beta, \boldsymbol{\rho}), \tag{5.6}$$

and discard variables that are not of interest. As previously discussed, π is not absolutely continuous with respect to the same base measure, though we may still decompose it as

$$\pi(\boldsymbol{n}, \boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{z}, \mathbf{u}) = \frac{1}{p(\mathcal{D} | \boldsymbol{\alpha}, \beta, \boldsymbol{\rho})} p(\boldsymbol{n} | \boldsymbol{\lambda}) p(\boldsymbol{\lambda} | \boldsymbol{\alpha}, \beta) p(\boldsymbol{c} | \boldsymbol{n}) p(\boldsymbol{\theta} | \boldsymbol{n}, \boldsymbol{\rho}) p(\mathbf{u}) p(\mathbf{z} | \boldsymbol{c}, \boldsymbol{\theta}) p(\mathcal{D} | \mathbf{z}, \mathbf{u}), \tag{5.7}$$

where we use the notation $p(.)$ and $p(.|.)$ to denote probability measures rather than probability density functions or probability mass functions, and where product and scaling operations are usual measure operations. Before proceeding any further, we will introduce a slight re-parameterisation of Equation (5.7) that will improve the inference scheme.

Let $\mathbf{n}_a = (\dots, \#\{a_k^j\}_k, \dots)$ be the vector of the numbers of unique boundary times in all d input dimensions. We recall from our prior on f that

$$p(\mathbf{z} | \boldsymbol{c}, \boldsymbol{\theta}) = \prod_{j=1}^{d} p\left(z_{a_0^j}^j, z_{a_0^j}^{j\prime}\right) \prod_{k=1}^{\boldsymbol{n}_a[j]-1} p\left(z_{a_k^j}^j, z_{a_k^j}^{j\prime} \middle| z_{a_{k-1}^j}^j, z_{a_{k-1}^j}^{j\prime}\right), \tag{5.8}$$

where each factor in the decomposition above is a bivariate Gaussian density whose mean vector and covariance matrix is obtained from the partitions $\boldsymbol{c}$, the kernel hyper-parameters $\boldsymbol{\theta}$, and the kernel membership scheme described in section 5.2.3 and illustrated in Figure 5.1, and using Equations (4.6-4.7). Let ${}_k^j\mathbf{K}_{u;v}$ be the unconditional

covariance matrix between $(z_u^j, z_u^{j\prime})$ and $(z_v^j, z_v^{j\prime})$ as per the unconditional kernel structure driving the string defined on the interval $[a_k^j, a_{k+1}^j[$. Let $\Sigma_0^j := {}_0^j\mathbf{K}_{a_0^j;a_0^j}$ be the auto-covariance matrix of $\left(z_{a_0^j}^j, z_{a_0^j}^{j\prime}\right)$. Let

$$\Sigma_k^j := {}_k^j\mathbf{K}_{a_k^j;a_k^j} - {}_k^j\mathbf{K}_{a_k^j;a_{k-1}^j}\,{}_k^j\mathbf{K}^{-1}_{a_{k-1}^j;a_{k-1}^j}\,{}_k^j\mathbf{K}^T_{a_k^j;a_{k-1}^j}$$

be the covariance matrix of $\left(z_{a_k^j}^j, z_{a_k^j}^{j\prime}\right)$ given $\left(z_{a_{k-1}^j}^j, z_{a_{k-1}^j}^{j\prime}\right)$, and

$$M_k^j = {}_k^j\mathbf{K}_{a_k^j;a_{k-1}^j}\,{}_k^j\mathbf{K}^{-1}_{a_{k-1}^j;a_{k-1}^j}.$$

Finally, let $L_k^j := U_k^j (D_k^j)^{\frac{1}{2}}$ with $\Sigma_k^j = U_k^j D_k^j (U_k^j)^T$ the singular value decomposition (SVD) of Σ_k^j. We may choose to represent $\left(z_{a_0^j}^j, z_{a_0^j}^{j\prime}\right)$ as

$$\begin{bmatrix} z_{a_0^j}^j \\ z_{a_0^j}^{j\prime} \end{bmatrix} = L_0^j x_0^j, \tag{5.9}$$

and for $k > 0$ we may also choose to represent $\left(z_{a_k^j}^j, z_{a_k^j}^{j\prime}\right)$ as

$$\begin{bmatrix} z_{a_k^j}^j \\ z_{a_k^j}^{j\prime} \end{bmatrix} = M_k^j \begin{bmatrix} z_{a_{k-1}^j}^j \\ z_{a_{k-1}^j}^{j\prime} \end{bmatrix} + L_k^j x_k^j, \tag{5.10}$$

where $\{x_k^j\}$ are independent bivariate standard normal vectors. Equations (5.9-5.10) provide an equivalent representation. In effect, we recall that if $Z = M + LX$, where $X \sim \mathcal{N}(0, I)$ is a standard multivariate Gaussian, M is a real vector, and L is a real matrix, then $Z \sim \mathcal{N}(M, LL^T)$. Equations (5.9-5.10) result from applying this result to $\left(z_{a_0^j}^j, z_{a_0^j}^{j\prime}\right)$ and $\left(z_{a_k^j}^j, z_{a_k^j}^{j\prime}\right) \Big| \left(z_{a_{k-1}^j}^j, z_{a_{k-1}^j}^{j\prime}\right)$. We note that at training time, M_k^j and L_k^j only depend on kernel hyper-parameters. Denoting $\mathbf{x}$ the vector of all x_k^j, $\mathbf{x}$ is a so-called 'whitened' representation of $\mathbf{z}$, which we prefer for reasons we will discuss shortly. In the whitened representation, the target distribution π is re-parameterized as

$$\pi(\boldsymbol{n}, \boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}) = \frac{1}{p(\mathcal{D}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})} p(\boldsymbol{n}|\boldsymbol{\lambda}) p(\boldsymbol{\lambda}|\boldsymbol{\alpha}, \boldsymbol{\beta}) p(\boldsymbol{c}|\boldsymbol{n}) p(\boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho}) p(\mathbf{u}) p(\mathbf{x}) p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}), \tag{5.11}$$

where the dependency of the likelihood term to the partitions and the hyper-parameters stems from the need to recover $\mathbf{z}$ and subsequently $\mathbf{f}$ from $\mathbf{x}$ through Equations (5.9) and (5.10). The whitened representation Equation (5.11) has two primary advantages. Firstly, it is robust to ill-conditioning of Σ_k^j, which would typically occur when two adjacent boundary times are too close to each other. In the representation of Equation (5.7), as one needs to evaluate the density $p(\mathbf{z}|\boldsymbol{c}, \boldsymbol{\theta})$, ill-conditioning of Σ_k^j would result in numerical instabilities. In contrast, in the whitened representation, one needs to evaluate the density $p(\mathbf{x})$, which is that of i.i.d. standard Gaussians and as such can be evaluated robustly. Moreover, the SVD required to evaluate L_k^j is also robust to ill-conditioning of Σ_k^j, so that Equations (5.9) and (5.10) hold and can be robustly evaluated for degenerate Gaussians too. The second advantage of the whitened representation is that it improves mixing by establishing a link between kernel hyper-parameters and the likelihood.

Equation (5.11) allows us to cast our inference problem as a Bayesian model selection problem under a countable family of models indexed by $\boldsymbol{n} \in \mathbb{N}^d$, each defined on a different parameter subspace $\mathcal{C}_{\boldsymbol{n}}$, with cross-model normalising constant $p(\mathcal{D}|\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho})$, model probability driven by $p(\boldsymbol{n}|\boldsymbol{\lambda})p(\boldsymbol{\lambda}|\boldsymbol{\alpha}, \boldsymbol{\beta})$, model-specific prior $p(\boldsymbol{c}|\boldsymbol{n})p(\boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho})p(\mathbf{u})p(\mathbf{x})$, and likelihood $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})$. Critically, it can be seen from Equation (5.11) that the conditional probability distribution $\pi(\boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}|\boldsymbol{n})$ admits a density with respect to Lebesgue's measure on $\mathcal{C}_{\boldsymbol{n}}$.

Our setup is therefore analogous to that which motivated the seminal paper Green (1995), so that to sample from the posterior $\pi(\boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}, \boldsymbol{n})$ we may use any Reversible-Jump Metropolis-Hastings (RJ-MH) scheme satisfying detailed balance and dimension-matching as described in section 3.3 of Green (1995). To improve mixing of the Markov chain, we will alternate between a *between-models* RJ-MH update with target distribution $\pi(\boldsymbol{n}, \boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u})$, and a *within-model* MCMC-within-Gibbs sampler with target distribution $\pi(\boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u}|\boldsymbol{n})$. Constructing reversible-jump samplers by alternating between within-model sampling and between-models sampling is standard practice, and it is well-known that doing so yields a Markov chain that converges to the target distribution of interest (see Brooks et al., 2011, p. 50).

In a slight abuse of notation, in the following we might use the notations $p(.|.)$ and $p(.)$, which we previously used to denote probability measures, to refer to the corresponding probability density functions or probability mass functions.

5.2.5 Within-Model Updates

We recall from Equation (5.11) that $\boldsymbol{c}, \boldsymbol{\theta}, \boldsymbol{\lambda}, \mathbf{x}, \mathbf{u} | \mathcal{D}, \boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\rho}, \boldsymbol{n}$ has probability density function

$$p(\boldsymbol{n}|\boldsymbol{\lambda})p(\boldsymbol{\lambda}|\boldsymbol{\alpha}, \boldsymbol{\beta})p(\boldsymbol{c}|\boldsymbol{n})p(\boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho})p(\mathbf{u})p(\mathbf{x})p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}), \tag{5.12}$$

up to a normalising constant.

Updating $\boldsymbol{\lambda}$: By independence of the priors over $(\boldsymbol{\lambda}[j], \boldsymbol{n}[j])$, the distributions $\boldsymbol{\lambda}[j] \;\middle|\; \boldsymbol{n}[j]$ are also independent, so that the updates may be performed in *parallel*. Moreover, recalling that the prior number of change-points in the j-th input dimension is Poisson distributed with intensity $\boldsymbol{\lambda}[j]\,(b^j - a^j)$, and by conjugacy of the Gamma distribution to the Poisson likelihood, it follows that

$$\boldsymbol{\lambda}[j] \;\middle|\; \boldsymbol{n}[j] \sim \Gamma\left(\frac{\boldsymbol{n}[j]}{b^j - a^j} + \boldsymbol{\alpha}[j], 1 + \boldsymbol{\beta}[j]\right). \tag{5.13}$$

This update step has memory requirement and time complexity both constant in the number of training and test samples.

Updating $\mathbf{u}$: When the likelihood has additional parameters $\mathbf{u}$, they may be updated with a Metropolis-Hastings step. Denoting $q(\mathbf{u} \to \mathbf{u}')$ the proposal probability density function, the acceptance ratio reads

$$r_{\mathbf{u}} = \min\left(1, \frac{p(\mathbf{u}')p\left(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}'\right) q(\mathbf{u}' \to \mathbf{u})}{p(\mathbf{u})p\left(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}\right) q(\mathbf{u} \to \mathbf{u}')}\right). \tag{5.14}$$

In some cases however, it might be possible and more convenient to choose $p(\mathbf{u})$ to be conjugate to the likelihood $p\left(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}\right)$. For instance, in regression problems under a Gaussian noise model, we may take $\mathbf{u}$ to be the noise variance on which we may place an inverse-gamma prior. Either way, the computational bottleneck of this step is the evaluation of the likelihood $p\left(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}'\right)$, which in most cases can be done with a time complexity and memory requirement that are both linear in the number of training samples.

Updating $\boldsymbol{c}$: We update the positions of change-points sequentially using the Metropolis-Hastings algorithm, one input dimension j at a time, and for each input dimension we proceed in increasing order of change-points. The proposal new position for the change-point c_p^j is sampled uniformly at random on the interval $]c_{p-1}^j, c_{p+1}^j[$, where c_{p-1}^j (resp. c_{p+1}^j) is replaced by a^j (resp. b^j) for the first (resp. last) change-point.

The acceptance probability of this proposal is easily found to be

$$r_{c_p^j} = \min \left(1, \frac{p\left(\mathcal{D}|\mathbf{x}, \boldsymbol{c}', \boldsymbol{\theta}, \mathbf{u}\right)}{p\left(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}\right)}\right), \tag{5.15}$$

where $\boldsymbol{c}'$ is identical to $\boldsymbol{c}$ except for the change-point to update. This step requires computing the factors $\{L_k^j, M_k^j\}$ corresponding to inputs in j-th dimension whose kernel configuration would change if the proposal were to be accepted, the corresponding vector of *derivative string GP* values $\mathbf{z}$, and the observation likelihood under the proposal $p\left(\mathcal{D}|\mathbf{x}, \boldsymbol{c}', \boldsymbol{\theta}, \mathbf{u}\right)$. The computational bottleneck of this step is therefore once again the evaluation of the new likelihood $p\left(\mathcal{D}|\mathbf{x}, \boldsymbol{c}', \boldsymbol{\theta}, \mathbf{u}\right)$.

Updating $\mathbf{x}$: The target conditional density of $\mathbf{x}$ is proportional to

$$p(\mathbf{x})p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}). \tag{5.16}$$

Recalling that $p(\mathbf{x})$ is a multivariate standard normal, it follows that the form of Equation (5.16) makes it convenient to use Elliptical Slice Sampling (Murray et al. (2010)) to sample from the unormalized conditional $p(\mathbf{x})p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})$. The two bottlenecks of this update step are sampling a new proposal from $p(\mathbf{x})$ and evaluating the likelihood $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})$. Sampling from the multivariate standard normal $p(\mathbf{x})$ may be *massively parallelized*, for instance by using GPU Gaussian random number generators. When no parallelism is available, the overall time complexity reads $\mathcal{O}\left(\sum_{j=1}^d \mathbf{n}_a[j]\right)$, where we recall that $\mathbf{n}_a[j]$ denotes the number of distinct training and testing input coordinates in the j-th dimension. In particular, if we denote N the total number of training and testing d-dimensional input samples, then $\sum_{j=1}^d \mathbf{n}_a[j] \leq dN$, although for many classes of datasets with sparse input values such as images, where each input (single-colour pixel value) may have at most 256 distinct values, we might have $\sum_{j=1}^d \mathbf{n}_a[j] \ll dN$. As for the memory required to sample from $p(\mathbf{x})$, it grows proportionally to the size of $\mathbf{x}$, that is in $\mathcal{O}\left(\sum_{j=1}^d \mathbf{n}_a[j]\right)$. In regards to the evaluation of the likelihood $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})$, as previously discussed its resource requirements are application-specific, but it will typically have time complexity that grows in $\mathcal{O}\left(N\right)$ and memory requirement that grows in $\mathcal{O}\left(dN\right)$. For instance, the foregoing resource requirements always hold for i.i.d. observation models such as in nonparametric regression and nonparametric classification problems.

Updating $\boldsymbol{\theta}$: We note from Equation (5.12) that the conditional distribution of $\boldsymbol{\theta}$ given everything else has unormalized density

$$p(\boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho})p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u}), \tag{5.17}$$

which we may choose to represent as

$$p(\log \boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho})p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \log \boldsymbol{\theta}, \mathbf{u}). \tag{5.18}$$

As we have put independent log-normal priors on the coordinates of $\boldsymbol{\theta}$ (see Equation (5.3)), we may once again use Elliptical Slice Sampling to sample from $\log \boldsymbol{\theta}$ before taking the exponential. The time complexity of generating a new sample from $p(\log \boldsymbol{\theta}|\boldsymbol{n}, \boldsymbol{\rho})$ will typically be at most linear in the total number of distinct kernel hyper-parameters. Overall, the bottleneck of this update is the evaluation of the likelihood $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \log \boldsymbol{\theta}, \mathbf{u})$. In this update, the latter operation requires recomputing the factors M_k^j and L_k^j of Equations (5.9) and (5.10), which requires computing and taking the SVD of unrelated 2×2 matrices, computations we may perform in *parallel*. Once the foregoing factors have been computed, we evaluate $\mathbf{z}$, the *derivative string GP* values at boundary times, parallelising over input dimensions, and running a sequential update within an input dimension using Equations (5.9) and (5.10). Updating $\mathbf{z}$ therefore has time complexity that is, in the worst case where no distributed computing is available, $\mathcal{O}(dN)$, and $\mathcal{O}(N)$ when there are up to d computing cores. The foregoing time complexity will also be that of this update step, unless the observation likelihood is more expensive. The memory requirement, as in previous updates, is $\mathcal{O}(dN)$.

Overall resource requirement: To summarize previous remarks, the overall computational bottleneck of a *within-model* iteration is the evaluation of the likelihood $p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})$. For i.i.d. observation models such as classification and regression problems for instance, the corresponding time complexity grows in $\mathcal{O}(N)$ when d computing cores are available, or $\mathcal{O}(dN)$ otherwise, and the memory requirement grows in $\mathcal{O}(dN)$.

5.2.6 Between-Models Updates

Our reversible-jump Metropolis-Hastings update proceeds as follows. We choose an input dimension, say j, uniformly at random. If j has no change-points, that is $\mathbf{n}[j] = 0$, we randomly choose between not doing anything, and adding a change-point, each outcome having the same probability. If $\mathbf{n}[j] > 0$, we either do nothing, add a change-point, or delete a change-point, each outcome having the same probability of occurrence.

Whenever we choose not to do anything, the acceptance ratio is easily found to be one:

$$r_{j0} = 1. \tag{5.19}$$

Whenever we choose to add a change-point, we sample the position c_*^j of the proposal new change-point uniformly at random on the domain $[a^j, b^j]$ of the j-th input dimension. This proposal will almost surely break an existing kernel membership cluster, say the p-th, into two; that is $c_p^j < c_*^j < c_{p+1}^j$ where we may have $a^j = c_p^j$ and/or $b^j = c_{p+1}^j$. In the event c_*^j coincides with an existing change-point, which should happen with probability 0, we do nothing. When adding a change-point, we sample a new vector of hyper-parameters θ_*^j from the log-normal prior of Equation (5.3), and we propose as hyper-parameters for the tentative new clusters $[c_p^j, c_*^j[$ and $[c_*^j, c_{p+1}^j[$ the vectors $\theta_{\text{add-left}}^j$ and $\theta_{\text{add-right}}^j$ defined as

$$\log \theta_{\text{add-left}}^j := \cos(\alpha) \log \theta_p^j - \sin(\alpha) \log \theta_*^j \tag{5.20}$$

and

$$\log \theta_{\text{add-right}}^j := \sin(\alpha) \log \theta_p^j + \cos(\alpha) \log \theta_*^j \tag{5.21}$$

respectively, where $\alpha \in [0, \frac{\pi}{2}]$ and θ_p^j is the vector of hyper-parameters currently driving the kernel membership defined by the cluster $[c_p^j, c_{p+1}^j[$. We note that if θ_p^j is distributed as per the prior in Equation (5.3) then $\theta_{\text{add-left}}^j$ and $\theta_{\text{add-right}}^j$ are i.i.d. distributed as per the foregoing prior. More generally, this elliptical transformation determines the extent to which the new proposal kernel configurations should deviate from the current configuration θ_p^j. α is restricted to $[0, \frac{\pi}{2}]$ so as to give a positive weight the the current vector of hyper-parameters θ_p^j. When $\alpha = 0$, the left hand-side cluster $[c_p^j, c_*^j[$ will fully exploit the current kernel configuration, while the right hand-side cluster $[c_*^j, c_{p+1}^j[$ will use the prior to explore a new set of hyper-parameters. When $\alpha = \frac{\pi}{2}$ the reverse occurs. To preserve symmetry between the left and right hand-side kernel configurations, we choose

$$\alpha = \frac{\pi}{4}. \tag{5.22}$$

Whenever we choose to delete a change-point, we choose an existing change-point uniformly at random, say c_p^j. Deleting c_p^j, would merge the clusters $[c_{p-1}^j, c_p^j[$ and $[c_p^j, c_{p+1}^j[$, where we may have $a^j = c_{p-1}^j$ and/or $b^j = c_{p+1}^j$. We propose as vector of hyper-parameters for the tentative merged cluster $[c_{p-1}^j, c_{p+1}^j[$ the vector $\theta_{\text{del-merged}}^j$ satisfying:

$$\log \theta_{\text{del-merged}}^j = \cos(\alpha) \log \theta_{p-1}^j + \sin(\alpha) \log \theta_p^j, \tag{5.23}$$

which together with

$$\log \theta^j_{\text{del-}*} = -\sin(\alpha) \log \theta^j_{p-1} + \cos(\alpha) \log \theta^j_p, \tag{5.24}$$

constitute the inverse of the transformation defined by Equations (5.20) and (5.21).

Whenever a proposal to add or delete a change-point occurs, the factors L^j_k and M^j_k that would be affected by the change in kernel membership structure are recomputed, and so are the affected coordinates of $\mathbf{z}$.

This scheme satisfies the reversibility and dimension-matching requirements of Green (1995). Moreover, the absolute value of the Jacobian of the mapping

$$\left(\log \theta^j_p, \log \theta^j_*\right) \to \left(\log \theta^j_{\text{add-left}}, \log \theta^j_{\text{add-right}}\right)$$

of the move to add a change-point in $[c^j_p, c^j_{p+1}[$ reads

$$\left| \frac{\partial \left(\log \theta^j_{\text{add-left}}, \log \theta^j_{\text{add-right}}\right)}{\partial \left(\log \theta^j_p, \log \theta^j_*\right)} \right| = 1. \tag{5.25}$$

Similarly, the absolute value of the Jacobian of the mapping corresponding to a move to delete change-point c^j_p, namely

$$\left(\log \theta^j_{p-1}, \log \theta^j_p\right) \to \left(\log \theta^j_{\text{del-merged}}, \log \theta^j_{\text{del-}*}\right),$$

reads:

$$\left| \frac{\partial \left(\log \theta^j_{\text{del-merged}}, \log \theta^j_{\text{del-}*}\right)}{\partial \left(\log \theta^j_{p-1}, \log \theta^j_p\right)} \right| = 1. \tag{5.26}$$

Applying the standard result Equation (8) of Green (1995), the acceptance ratio of the move to add a change-point is found to be

$$r_{j+} = \min\left(1, \frac{p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}_+, \boldsymbol{\theta}_+, \mathbf{u})}{p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})} \frac{\boldsymbol{\lambda}[j]\,(b^j - a^j)}{1 + \mathbf{n}[j]} \frac{p_{\log \theta^j}\left(\log \theta^j_{\text{add-left}}\right) p_{\log \theta^j}\left(\log \theta^j_{\text{add-right}}\right)}{p_{\log \theta^j}\left(\log \theta^j_p\right) p_{\log \theta^j}\left(\log \theta^j_*\right)}\right) \tag{5.27}$$

where $p_{\log \theta^j}$ is the prior over log hyper-parameters in the j-th input dimension (as per the prior specification Equation (5.3)), which we recall is i.i.d. centred Gaussian with variance $\boldsymbol{\rho}[j]$, and $\boldsymbol{c}_+$ and $\boldsymbol{\theta}_+$ denote the proposal new vector of change-points and the corresponding vector of hyper-parameters. The three coloured terms in the acceptance probability are very intuitive. The green term $\frac{p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}_+, \boldsymbol{\theta}_+, \mathbf{u})}{p(\mathcal{D}|\mathbf{x}, \boldsymbol{c}, \boldsymbol{\theta}, \mathbf{u})}$ represents the

fit improvement that would occur if the new proposal is accepted. In the red term $\frac{\boldsymbol{\lambda}[j]\left(b^j-a^j\right)}{1+\mathbf{n}[j]}$, $\boldsymbol{\lambda}[j]\,(b^j-a^j)$ represents the average number of change-points in the j-th input dimension as per the HPP prior, while $1+\mathbf{n}[j]$ corresponds to the proposed new number of change-points in the j-th dimension, so that the whole red term acts as a complexity regulariser. Finally, the blue term $\frac{p_{\log\theta^j}\left(\log\theta^j_{\text{add-left}}\right)p_{\log\theta^j}\left(\log\theta^j_{\text{add-right}}\right)}{p_{\log\theta^j}\left(\log\theta^j_{p}\right)p_{\log\theta^j}\left(\log\theta^j_{*}\right)}$ plays the role of a hyper-parameter regulariser.

Similarly, the acceptance ratio of the move to delete change-point c^j_p, thereby changing the number of change-points in the j-th input dimension from $\mathbf{n}[j]$ to $\mathbf{n}[j]-1$, is found to be

$$r_{j-}=\min\left(1,\frac{p(\mathcal{D}|\mathbf{x},\boldsymbol{c}_-,\boldsymbol{\theta}_-,\mathbf{u})}{p(\mathcal{D}|\mathbf{x},\boldsymbol{c},\boldsymbol{\theta},\mathbf{u})}\frac{\mathbf{n}[j]}{\boldsymbol{\lambda}[j]\,(b^j-a^j)}\frac{p_{\log\theta^j}\left(\log\theta^j_{\text{del-merged}}\right)p_{\log\theta^j}\left(\log\theta^j_{\text{del-}*}\right)}{p_{\log\theta^j}\left(\log\theta^j_{p-1}\right)p_{\log\theta^j}\left(\log\theta^j_{p}\right)}\right), \tag{5.28}$$

where $\boldsymbol{c}_-$ and $\boldsymbol{\theta}_-$ denote the proposal new vector of change-points and the corresponding vector of hyper-parameters. Once again, each coloured term plays the same intuitive role as its counterpart in Equation (5.27).

Overall resource requirement: The bottleneck of between-models updates is the evaluation of the new likelihoods $p(\mathcal{D}|\mathbf{x},\boldsymbol{c}_+,\boldsymbol{\theta}_+,\mathbf{u})$ or $p(\mathcal{D}|\mathbf{x},\boldsymbol{c}_-,\boldsymbol{\theta}_-,\mathbf{u})$, whose resource requirements, which are the same as those of within-models updates, we already discussed.

Algorithm 5.1 summarises the proposed MCMC sampler.

Algorithm 5.1 MCMC sampler for nonparametric Bayesian inference of a real-valued latent function under a *string GP* prior

Inputs: Likelihood model $p(\mathcal{D}|\mathbf{f},\mathbf{u})$, link function ϕ, training data $\mathcal{D}$, test inputs, type of unconditional kernel, prior parameters $\boldsymbol{\alpha},\boldsymbol{\beta},\boldsymbol{\rho}$.
Outputs: Posterior samples of the values of the latent function at training and test inputs $\mathbf{f}$ and $\mathbf{f}^*$, and the corresponding gradients $\nabla\mathbf{f}$ and $\nabla\mathbf{f}^*$.

Step 0: Initialize the chain with no change-point ($\boldsymbol{n}=0$ and $\boldsymbol{c}=\emptyset$), and sample the remaining variables $\boldsymbol{\theta},\boldsymbol{\lambda},\mathbf{x},\mathbf{u}$ from their priors.
repeat
Step 1: Perform a within-model update.
1.1: Update each $\boldsymbol{\lambda}[j]$ by sampling from the Gamma distribution in Equation (5.13).
1.2: Update $\mathbf{u}$, the vector of other likelihood parameters, if any, using Metropolis-Hastings (MH) with proposal q and acceptance ratio Equation (5.14) or by sampling directly from the posterior when $p(\mathbf{u})$ is conjugate to the likelihood model.
1.3: Update $\boldsymbol{\theta}$, using Elliptical Slice Sampling (ESS) with target distribution Equation (5.18), and record the newly computed factors $\{L_k^j, M_k^j\}$ that relate $\mathbf{z}$ to its whitened representation $\mathbf{x}$.
1.4: Update $\mathbf{x}$ using ESS with target distribution Equation (5.16).
1.5: Update change-point positions $\boldsymbol{c}$ sequentially using MH, drawing a proposal update for c_p^j uniformly at random on $]c_{p-1}^J, c_{p+1}^j[$, and accepting the update with probability $r_{c_p^j}$ (defined Equation (5.15)). On accept, update the factors $\{L_k^j, M_k^j\}$.
Step 2: Perform a between-models update.
2.1: Sample a dimension to update, say j, uniformly at random.
2.2: Consider adding or deleting a change-point
if $\mathbf{n}[j]=0$ **then**
Randomly choose to add a change-point with probability 1/2.
if we should consider adding a change-point **then**
Construct proposals to update following section 5.2.6.
Accept proposals with probability r_+^j (see Equation (5.27)).
end if
else
Randomly choose to add/delete a change-point with probability 1/3.
if we should consider adding a change-point **then**
Construct proposals to update following section 5.2.6.
Accept proposals with probability r_+^j (see Equation (5.27)).
else if we should consider deleting a change-point **then**
Construct proposals to update following section 5.2.6.
Accept proposals with probability r_-^j (see Equation (5.28)).
else
Continue.
end if
end if
Step 3: Compute $\mathbf{f},\mathbf{f}^*,\nabla\mathbf{f}$ and $\nabla\mathbf{f}^*$, first recovering $\mathbf{z}$ from $\mathbf{x}$, and then recalling that $f(x)=\phi\left(z^1_{x[1]},\ldots,z^d_{x[d]}\right)$ and $\nabla f(x)=\left(z^{1\prime}_{x[1]}\frac{\partial\phi}{\partial x[1]}(x),\ldots,z^{d\prime}_{x[d]}\frac{\partial\phi}{\partial x[d]}(x)\right)$.
until enough samples are generated after mixing.

5.3 Multi-Output Problems

Although we have restricted ourselves to cases where the likelihood model depends on a single real-valued function for brevity and to ease notations, cases where the likelihood depends on vector-valued functions, or equivalently multiple real-valued functions, present no additional theoretical or practical challenge. We may simply put independent *string GP* priors on each of the latent functions. An MCMC sampler almost identical to the one introduced herein may be used to sample from the posterior. All that is required to adapt the proposed MCMC sampler to multi-outputs problems is to redefine $\mathbf{z}$ to include all univariate *derivative string GP* values across input dimensions and across latent functions, perform step 1.1 of Algorithm 5.1 for each of the latent function, and update step 2.1 so as to sample uniformly at random not only what dimension to update but also what latent function. Previous analyses and derived acceptance ratios remain unchanged. The resource requirements of the resulting multi-outputs MCMC sampler on a problem with K latent functions, N training and test d-dimensional inputs, are the same as those of the MCMC sampler for a single output (Algorithm 5.1) with N training and test dK-dimensional inputs. The time complexity is $\mathcal{O}(N)$ when dK computing cores are available, $\mathcal{O}(dKN)$ when no distributed computing is available, and the memory requirement becomes $\mathcal{O}(dKN)$.

5.4 Flashback to Small Scale GP Regression with String GP Kernels

In section 5.1 we discussed maximum marginal likelihood inference in Bayesian non-parametric regression under additively separable *string GP* priors, or GP priors with *string GP* covariance functions. We proposed learning the positions of boundary times, conditional on their number, jointly with kernel hyper-parameters and noise variances by maximising the marginal likelihood using gradient-based techniques. We then suggested learning the number of strings in each input dimension by trading off goodness-of-fit with model simplicity using information criteria such as AIC and BIC. In this section we propose a fully Bayesian nonparametric alternative.

Let us consider the Gaussian process regression model

$$y_i = f(x_i) + \epsilon_i, \quad f \sim \mathcal{GP}\left(0, k_{\text{SGP}}(.,.)\right), \quad \epsilon_i \sim \mathcal{N}\left(0, \sigma^2\right), \tag{5.29}$$

$$x_i \in [a^1, b^1] \times \cdots \times [a^d, b^d], \quad y_i, \epsilon_i \in \mathbb{R}, \tag{5.30}$$

where k_{SGP} is the covariance function of some *string GP* with boundary times $\{a_k^j\}$ and corresponding unconditional kernels $\{k_k^j\}$ in the j-th input dimension. It is worth stressing that we place a GP (not *string GP*) prior on the latent function f, but the covariance function of the GP is a *string GP* covariance function (as discussed in section 4.4.2 and as derived in Appendix C.8). Of course when the *string GP* covariance function k_{SGP} is separately additive, the two functional priors are the same. However, we impose no restriction on the link of the *string GP* that k_{SGP} is the covariance function of, other than continuous differentiability. To make full Bayesian nonparametric inference, we may place independent homogeneous Poisson process priors on the boundary times $\{a_k^j\}$, each with intensity λ^j. Similarly to the previous section (Equation (5.3)) our full prior specification of the *string GP* kernel reads

$$\begin{cases} \lambda^j \sim \Gamma(\alpha^j, \beta^j), \\ \{a_k^j\} \Big| \lambda^j \sim \text{HPP}(\lambda^j) \\ \theta_k^j[i] \Big| \{a_k^j\}, \lambda^j \overset{\text{i.i.d}}{\sim} \log \mathcal{N}(0, \rho^j) \\ \forall (j,k) \neq (l,p)\ \theta_k^j \perp \theta_p^l, \end{cases}, \tag{5.31}$$

where θ_k^j is the vector of hyper-parameters driving the unconditional kernel k_k^j. The method developed in the previous section and the resulting MCMC sampling scheme (Algorithm 5.1) may be reused to sample from the posterior over function values, pending the following two changes. First, gradients $\nabla \mathbf{f}$ and $\nabla \mathbf{f}^*$ are no longer necessary. Second, we may work with function values $(\mathbf{f}, \mathbf{f}^*)$ directly (that is in the original as opposed to whitened space). The resulting (Gaussian) distribution of function values $(\mathbf{f}, \mathbf{f}^*)$ conditional on all other variables is then analytically derived using standard Gaussian identities, like it is done in vanilla Gaussian process regression, so that the within-model update of $(\mathbf{f}, \mathbf{f}^*)$ is performed using a single draw from a multivariate Gaussian.

This approach to model complexity learning is advantageous over the information criteria alternative of section 5.1 in that it scales better with large input-dimensions. Indeed, rather than performing complete maximum marginal likelihood inference a number of times that grows exponentially with the input dimension, the approach of this section alternates between exploring a new combination of numbers of kernel configurations in each input dimension, and exploring function values and kernel hyper-parameters (given their number). That being said, this approach should only be considered as an alternative to commonly used kernels for *small scale* regression

problems to enable the learning of local patterns. Crucially, it scales as poorly as the *standard GP paradigm*, and Algorithm 5.1 should be preferred for large scale problems.

5.5 Experiments

We now move on to presenting empirical evidence for the efficacy of *string GPs* in coping with local patterns in datasets, and in doing so in a scalable manner. Firstly we consider maximum marginal likelihood inference on two small scale problems exhibiting local patterns. We begin with a toy experiment that illustrates the limitations of the *standard GP paradigm* in extrapolating and interpolating simple local periodic patterns. Then, we move on to comparing the accuracy of Bayesian nonparametric regression under a *string GP* prior to that of the standard Gaussian process regression model and existing mixture-of-experts alternatives on the motorcycle dataset of Silverman (1985), commonly used for the local patterns it exhibits. Finally, we illustrate the performance of the previously derived MCMC sampler on two large scale Bayesian inference problems, namely the prediction of U.S. commercial airline arrival delays of Hensman et al. (2013) and a new large scale dynamic asset allocation problem.

5.5.1 Extrapolation and Interpolation of Synthetic Local Patterns

In our first experiment, we illustrate a limitation of the standard approach consisting of postulating a global covariance structure on the domain, namely that this approach might result in unwanted global extrapolation of local patterns, and we show that this limitation is addressed by the *string GP paradigm*. To this aim, we use two toy regression problems. We consider the following functions:

$$f_0(t) = \begin{cases} \sin(60\pi t) & t \in [0, 0.5] \\ \frac{15}{4}\sin(16\pi t) & t \in]0.5, 1] \end{cases}, \quad f_1(t) = \begin{cases} \sin(16\pi t) & t \in [0, 0.5] \\ \frac{1}{2}\sin(32\pi t) & t \in]0.5, 1] \end{cases}. \tag{5.32}$$

f_0 (resp. f_1) undergoes a sharp (resp. mild) change in frequency and amplitude at $t = 0.5$. We consider using their restrictions to $[0.25, 0.75]$ for training. We sample those restrictions with frequency 300, and we would like to extrapolate the functions to the rest of their domains using Bayesian nonparametric regression.

We compare marginal likelihood *string GP* regression models, as described in section 5.1, to vanilla GP regression models using popular and expressive kernels. All *string GP* models have two strings and the partition is learned in the marginal likelihood

maximisation. Figure 5.2 illustrates plots of the posterior means for each kernel used, and Table 5.1 compares predictive errors. Overall, it can be noted that the *string GP kernel* with the periodic kernel (MacKay (1998)) as building block outperforms competing kernels, including the expressive spectral mixture kernel

$$k_{\mathrm{SM}}(r) = \sum_{k=1}^{K} \sigma_k^2 \exp(-2\pi^2 r^2 \gamma_k^2) \cos(2\pi r \mu_k)$$

of Wilson and Adams (2013) with $K = 5$ mixture components.[6]

The comparison between the spectral mixture kernel and the string spectral mixture kernel is of particular interest, since spectral mixture kernels are pointwise dense in the family of stationary kernels, and thus can be regarded as flexible enough for learning *stationary* kernels from the data. In our experiment, the string spectral mixture kernel with a single mixture component per string significantly outperforms the spectral mixture kernel with 5 mixture components. This intuitively can be attributed to the fact that, regardless of the number of mixture components in the spectral mixture kernel, the learned kernel must account for both types of patterns present in each training dataset. Hence, each local extrapolation on each side of 0.5 will attempt to make use of both amplitudes and both frequencies evidenced in the corresponding training dataset, and will struggle to recover the true local sine function. We would expect that the performance of the spectral mixture kernel in this experiment will not improve drastically as the number of mixture components increases. However, under a *string GP* prior, the left and right hand side strings are independent conditional on the (unknown) boundary conditions. Therefore, when the *string GP* domain partition occurs at time 0.5, the training dataset on $[0.25, 0.5]$ influences the hyper-parameters of the string to the right of 0.5 only to the extent that both strings should agree on the value of the latent function and its derivative at 0.5. To see why this is a weaker condition, we consider the family of pair of functions:

$$(\alpha\omega_1 \sin(\omega_2 t),\ \alpha\omega_2 \sin(\omega_1 t)), \quad \omega_i = 2\pi k_i, \quad k_i \in \mathbb{N}, \quad \alpha \in \mathbb{R}.$$

Such functions always have the same value and derivative at 0.5, regardless of their frequencies, and they are plausible GP paths under a spectral mixture kernel with one single mixture component ($\mu_k = k_i$ and $\gamma_k \ll 1$), and under a periodic kernel. As such

[6] The sparse spectrum kernel of Lazaro-Gredilla et al. (2010) can be thought of as the special case $\gamma_k \ll 1$.

it is not surprising that extrapolation under a string spectral mixture kernel or a string periodic kernel should perform well.

To further illustrate that *string GPs* are able to learn local patterns that GPs with commonly used and expressive kernels can't, we consider interpolating two bivariate functions f_2 and f_3 that exhibit local patterns. The functions are defined as:

$$\forall u, v \in [0.0, 1.0] \quad f_2(u, v) = f_0(u) f_1(v), \quad f_3(u, v) = \sqrt{f_0(u)^2 + f_1(v)^2}. \tag{5.33}$$

We consider recovering the original functions as the posterior mean of a GP regression model trained on $[0.0, 0.4] \cup [0.6, 1.0] \times [0.0, 0.4] \cup [0.6, 1.0]$. Each bivariate kernel used is a product of two univariate kernels in the same family, and we used standard Kronecker techniques to speed-up inference (see Saatchi, 2011, p.134). The univariate kernels we consider are the same as previously. Each univariate *string GP* kernel has one change-point (two strings) whose position is learned by maximum marginal likelihood. Results are illustrated in Figures 5.3 and 5.4. Once again it can be seen that unlike any competing kernel, the product of string periodic kernels recover both functions almost perfectly. In particular, it is impressive to see that, despite f_3 not being a separable function, a product of string periodic kernels recovered it almost perfectly. The interpolations performed by the spectral mixture kernel (see Figures 5.3 and 5.4) provide further evidence for our previously developed narrative: the spectral mixture kernel tries to blend all local patterns found in the training data during the interpolation. The periodic kernel learns a single global frequency characteristic of the whole dataset, ignoring local patterns, while the squared exponential, Matérn and rational quadratic kernels merely attempt to perform interpolation by smoothing.

Although we used synthetic data to ease illustrating our argument, it is reasonable to expect that in real-life problems the bigger the dataset, the more likely there might be local patterns that should not be interpreted as noise and yet are not indicative of the dataset as whole.

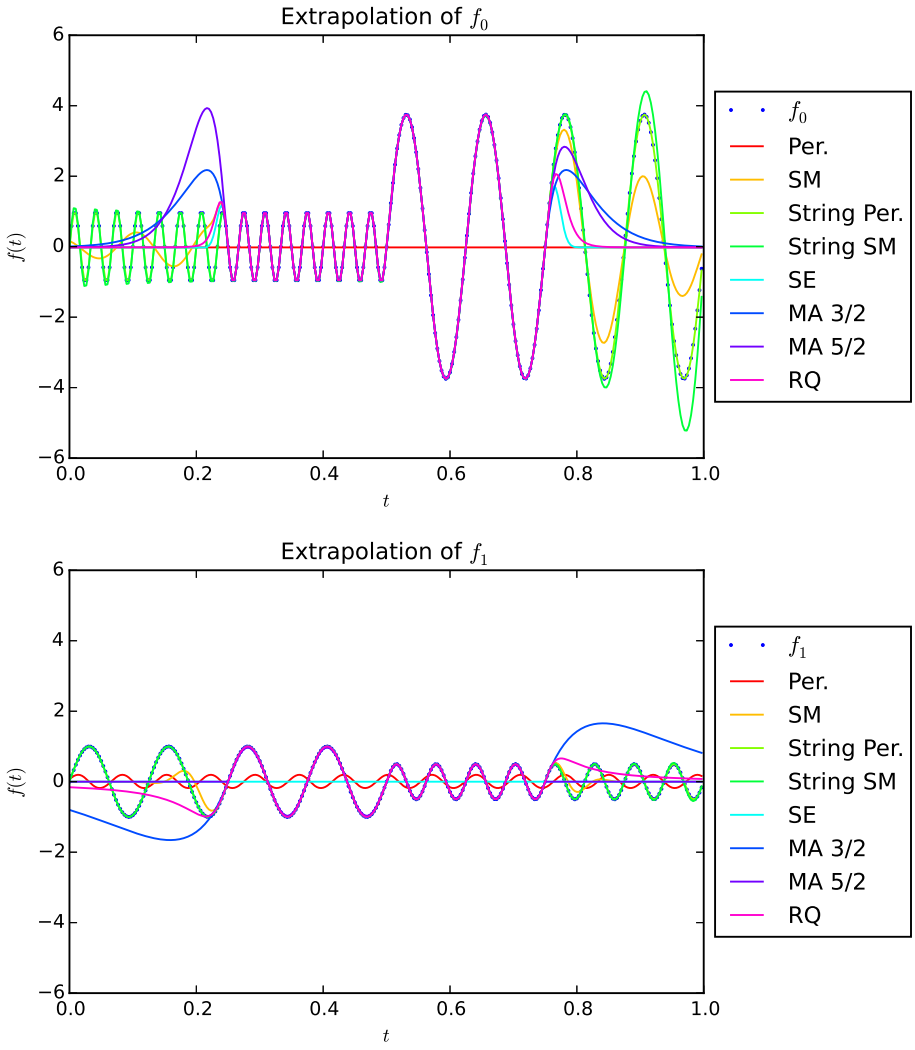

Fig. 5.2 Extrapolation of two functions f_0 and f_1 through Bayesian nonparametric regression under *string GP* priors and vanilla GP priors with popular and expressive kernels. Each model is trained on $[0.25, 0.5]$ and extrapolates to $[0, 1.0]$.

	Absolute Error		Squared Error	
Kernel	f_0	f_1	f_0	f_1
Squared exponential	1.44 ± 2.40	0.48 ± 0.58	3.50 ± 9.20	0.31 ± 0.64
Rational quadratic	1.39 ± 2.31	0.51 ± 0.83	3.28 ± 8.79	0.43 ± 1.15
Matérn 3/2	1.63 ± 2.53	1.26 ± 1.37	4.26 ± 11.07	2.06 ± 3.55
Matérn 5/2	1.75 ± 2.77	0.48 ± 0.58	5.00 ± 12.18	0.31 ± 0.64
Periodic	1.51 ± 2.45	0.53 ± 0.60	3.79 ± 9.62	0.37 ± 0.72
Spec. Mix. (5 comp.)	0.75 ± 1.15	0.39 ± 0.57	0.94 ± 2.46	0.24 ± 0.58
String Spec. Mix. (2 strings, 1 comp.)	0.23 ± 0.84	0.01 ± 0.03	0.21 ± 1.07	$\mathbf{0.00 \pm 0.00}$
String Periodic	$\mathbf{0.02 \pm 0.02}$	$\mathbf{0.00 \pm 0.01}$	$\mathbf{0.00 \pm 0.00}$	$\mathbf{0.00 \pm 0.00}$

Table 5.1 Predictive accuracies in the extrapolation of the two functions f_0 and f_1 of Section 5.5.1 through Bayesian nonparametric regression under *string GP* priors and vanilla GP priors with popular and expressive kernels. Each model is trained on $[0.25, 0.5]$ and extrapolates to $[0, 1.0]$. The predictive errors are reported as average $\pm$ 2 standard deviations.

5.5.2 Small Scale Heteroskedastic Regression

In our second experiment, we illustrate the advantage of the *string GP paradigm* over the *standard GP paradigm*, and also over the alternatives of Kim et al. (2005), Gramacy and Lee (2008), Tresp (2000) and Deisenroth and Ng (2015) that consist of considering independent GP experts on disjoint parts of the domain or handling disjoint subsets of the data. Using the motorcycle dataset of Silverman (1985), commonly used for the local patterns it exhibits, we show that our approach outperforms the aforementioned competing alternatives, thereby providing empirical evidence that the collaboration between consecutive GP experts introduced in the *string GP paradigm* vastly improves predictive accuracy and certainty in regression problems with local patterns. We also illustrate learning of the derivative of the latent function, solely from noisy measurements of the latent function.

The observations consist of accelerometer readings taken through time in an experiment on the efficacy of crash helmets. It can be seen at a glance in Figure 5.5 that the dataset exhibits roughly 4 regimes. Firstly, between 0ms and 15ms the acceleration was negligible. Secondly, the impact slowed down the helmet, resulting in a sharp deceleration between 15ms and 28ms. Thirdly, the helmet seems to have bounced back between 28ms and 32ms, before it finally gradually slowed down and came to a stop between 32ms and 60ms. It can also be noted that the measurement noise seems to have been higher in the second half of the experiment.

We ran 50 independent random experiments, leaving out 5 points selected uniformly at random from the dataset for prediction, the rest being used for training. The models we considered include the vanilla GP regression model, the *string GP* regression model with marginal maximum likelihood inference as described in section 5.1, mixtures of independent GP experts acting on disjoint subsets of the data both for training and testing, the Bayesian committee machine (Tresp (2000)), and the robust Bayesian committee machine (Deisenroth and Ng (2015)). We considered *string GPs* with 4 and 6 strings whose boundary times are learned as part of the maximum likelihood inference. For consistency, we used the resulting partitions of the domain to define the independent experts in the competing alternatives we considered. The Matérn 3/2 kernel was used throughout. The results are reported in Table 5.2. To gauge the ability of each model to capture the physics of the helmets crash experiment, we have also trained all models with all data points. The results are illustrated in Figures 5.5 and 5.6.

It can be seen at a glance from Figure 5.6 that mixtures of independent GP experts are inappropriate for this experiment as i) the resulting posterior means exhibit

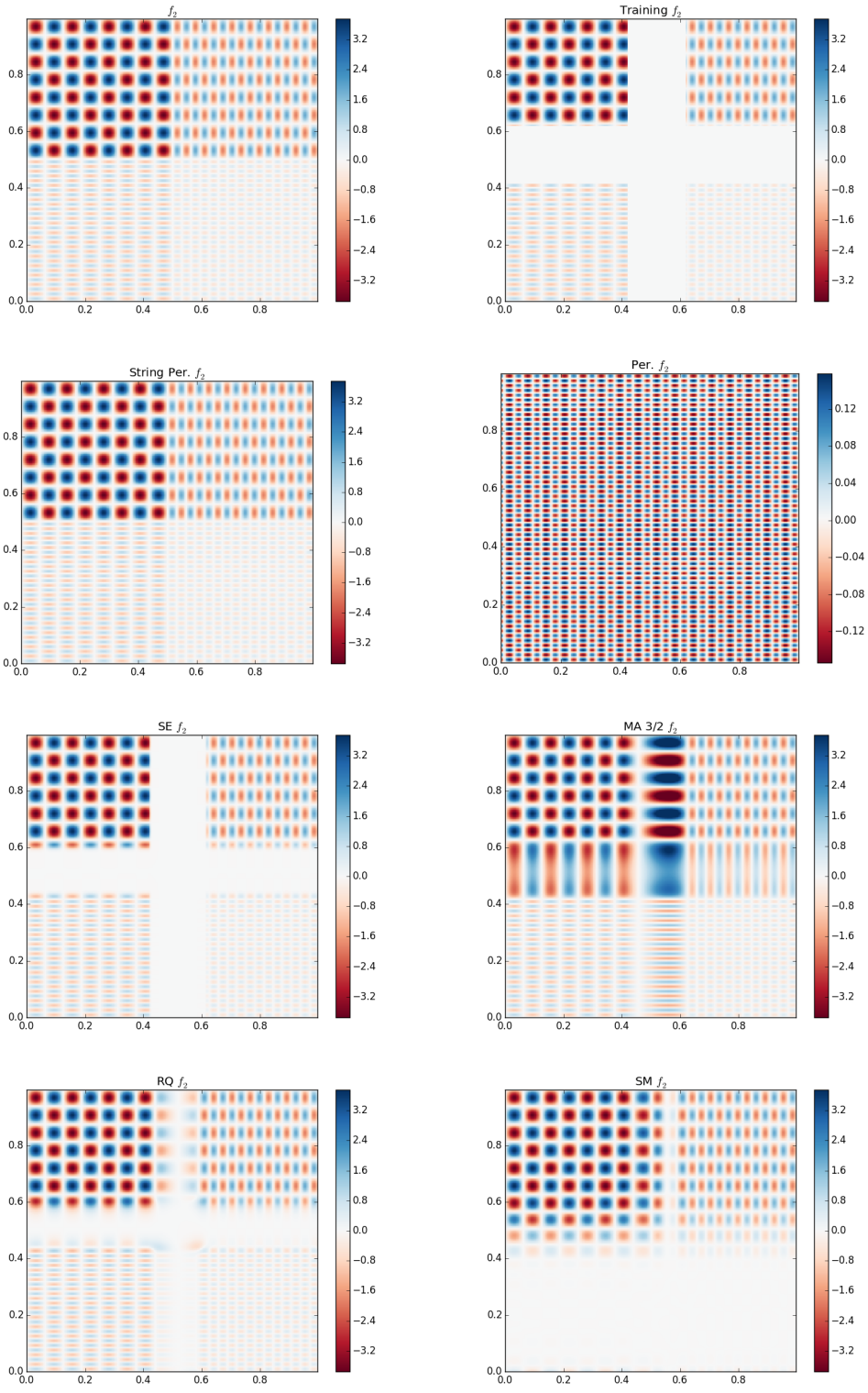

Fig. 5.3 Extrapolation of a synthetic function f_2 (top left corner), cropped in the middle for training (top right corner), using *string GP* regression and vanilla GP regression with various popular and expressive kernels.

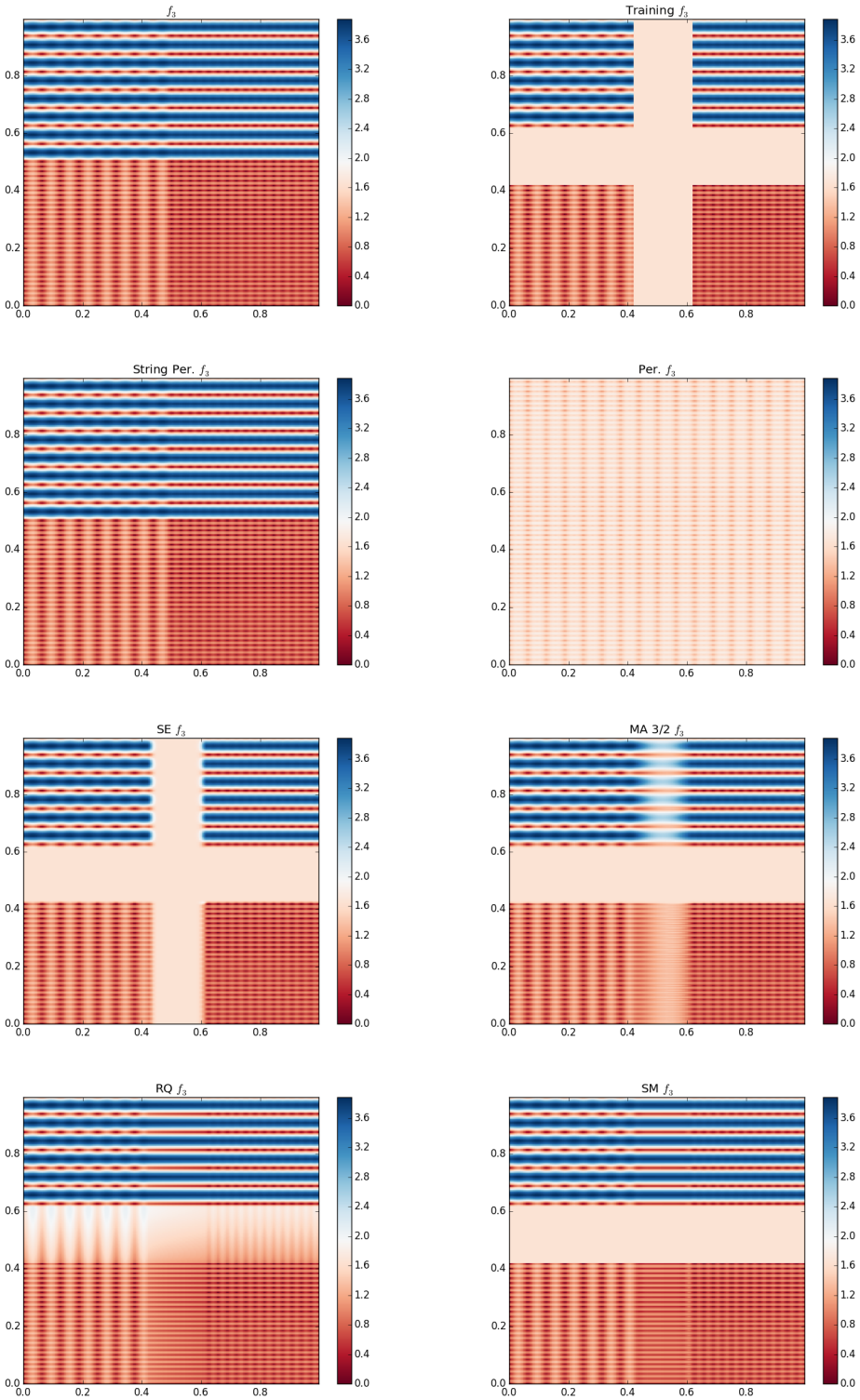

Fig. 5.4 Extrapolation of a synthetic function f_3 (top left corner), cropped in the middle for training (top right corner), using *string GP* regression and vanilla GP regression with various popular and expressive kernels.

discontinuities (for instance at $t = 30$ms and $t = 40$ms) that are inconsistent with the physics of the underlying phenomenon, and ii) they overfit the data towards the end. These foregoing discontinuities do not come as a surprise as each GP regression expert acts on a specific subset of the domain that is disjoint from the ones used by the other experts, both for training and prediction. Thus, there is no guarantee of consistency between expert predictions at the boundaries of the domain partition. Another perspective to this observation is found in noting that postulating independent GP experts, each acting on an element of a partition of the domain, is equivalent to putting as prior on the whole function a stochastic process that is discontinuous at the boundaries of the partition. Thus, the posterior stochastic process should not be expected to be continuous at the boundaries of the domain either.

This discontinuity issue is addressed by the Bayesian committee machine (BCM) and the robust Bayesian committee machine (rBCM) because, despite each independent expert being trained on a disjoint subset of the data, each expert is tasked with making predictions about all test inputs, not just the ones that fall into its input subspace. Each GP expert prediction is therefore continuous on the whole input domain,[7] and the linear weighting schemes operated by the BCM and the rBCM on expert predictions to construct the overall predictive mean preserve continuity. However, we found that the BCM and the rBCM suffer from three pitfalls. First, we found them to be less accurate than any other alternative out-of-sample on this dataset (see Table 5.2). Second, their predictions of latent function values are overly uncertain. This can be justified by the fact that, each GP expert being trained only with training samples that lie on its input subspace, its predictions about test inputs that lie farther away from its input subspace will typically be much more uncertain, so that, despite the weighting scheme of the Bayesian committee machine putting more mass on 'confident' experts, overall the posterior variance over latent function values might still be much higher than in the *standard GP paradigm* for instance. This is well illustrated by both the last column of Table 5.2 and the third and fourth rows of Figure 5.6. On the contrary, no *string GP* model suffers from this excess uncertainty problem. Third, the posterior means of the BCM, the rBCM and the vanilla GP regression exhibit oscillations towards the end ($t > 40$ms) that are inconsistent with the experimental setup; the increases in acceleration as the helmet slows down suggested by these posterior means would require an additional source of energy after the bounce.

In addition to being more accurate and more certain about predictions than vanilla GP regression, the BCM and the rBCM (see Table 5.2) , *string GP* regressions yield

[7] So long as the functional prior is continuous, which is the case here.

posterior mean acceleration profiles that are more consistent with the physics of the experiment: steady speed prior to the shock, followed by a deceleration resulting from the shock, a brief acceleration resulting from the change in direction after the bounce, and finally a smooth slow down due to the dissipation of kinetic energy. Moreover, unlike the vanilla GP regression, the rBCM and the BCM, *string GP* regressions yield smaller posterior variances towards the beginning and the end of the experiment than in the middle, which is consistent with the fact that the operator would be less uncertain about the acceleration at the beginning and at the end of the experiment—one would indeed expect the acceleration to be null at the beginning and at the end of the experiment. This desirable property can be attributed to the heteroskedasticity of the noise structure in the *string GP* regression model.

We also learned the derivative of the latent acceleration with respect to time, purely from noisy acceleration measurements using the joint law of a *string GP* and its derivative (Theorem 4.2). This is illustrated in Figure 5.5.

Table 5.2 Performance comparison between *string GPs*, vanilla GPs, mixture of independent GPs, the Bayesian committee machine (Tresp (2000)) and the robust Bayesian committee machine (Deisenroth and Ng (2015)) on the motorcycle dataset of Silverman (1985). The Matérn 3/2 kernel was used throughout. The domain partitions were learned in the *string GP* experiments by maximum likelihood. The learned partitions were then reused to allocate data between GP experts in other models. 50 random runs were performed, each run leaving 5 data points out for testing and using the rest for training. All results (except for predictive standard deviations) are reported as average over the 50 runs ± standard error. The last column contains the minimum, average and maximum of the predictive standard deviation of the values of the latent (noise-free) function at all test points across random runs.

	Training	Prediction			
	Log. lik.	Log. lik.	Absolute Error	Squared Error	Pred. Std
String GP (4 strings)	-388.36 ± 0.36	-22.16 ± 0.41	$\mathbf{15.70 \pm 1.05}$	$\mathbf{466.47 \pm 50.74}$	0.70/2.25/3.39
String GP (6 strings)	$\mathbf{-367.21 \pm 0.43}$	-21.99 ± 0.37	15.89 ± 1.06	475.59 ± 51.95	**0.64/2.21/3.46**
Vanilla GP	-420.69 ± 0.24	-22.77 ± 0.24	16.84 ± 1.09	524.18 ± 58.33	2.66/3.09/4.94
Mix. of 4 GPs	-388.37 ± 0.38	-20.90 ± 0.38	16.61 ± 1.10	512.30 ± 56.08	1.67/2.85/4.59
Mix. of 6 GPs	-369.05 ± 0.45	$\mathbf{-20.11 \pm 0.45}$	16.05 ± 1.11	500.43 ± 58.26	0.62/2.83/4.63
BCM with 4 GPs	-419.08 ± 0.30	-22.94 ± 0.26	17.17 ± 1.13	538.94 ± 61.91	7.20/9.92/22.92
BCM with 6 GPs	-422.15 ± 0.30	-22.91 ± 0.26	16.93 ± 1.12	533.21 ± 61.78	7.09/9.93/25.10
rBCM with 4 GPs	-419.08 ± 0.30	-22.99 ± 0.27	17.29 ± 1.11	546.95 ± 61.21	5.86/9.08/27.52
rBCM with 6 GPs	-422.15 ± 0.30	-22.96 ± 0.28	16.79 ± 1.12	542.95 ± 61.95	5.15/8.61/29.15

5.5.3 Large Scale Regression

To illustrate how our approach fares against competing alternatives on a standard large scale problem, we consider predicting the arrival delays of commercial flights in the USA in 2008 as studied by Hensman et al. (2013). We choose the same covariates as in Hensman et al. (2013), namely the age of the aircraft (number of years since deployment), distance that needs to be covered, airtime, departure time, arrival time, day of the week, day of the month and month. Unlike Hensman et al. (2013) who only considered commercial flights between January 2008 and April 2008, we consider commercial throughout the whole year, giving a total of 5.93 million records. In addition to the whole dataset, we also consider subsets so as to empirically illustrate the sensitivity of computational time to the number of samples. Selected subsets consist of $10,000$, $100,000$ and $1,000,000$ records selected uniformly at random. For each dataset, we use $2/3$ of the records selected uniformly at random for training and we use the remaining $1/3$ for testing. As competing alternatives to *string GPs* we consider the SVIGP of Hensman et al. (2013), the Bayesian committee machines (BCM) of Tresp (2000), and the robust Bayesian committee machines (rBCM) of Deisenroth and Ng (2015).

As previously discussed the prediction scheme operated by the BCM is Kolmogorov-inconsistent in that the resulting predictive distributions are not consistent by marginalization.[8] Moreover, jointly predicting all function values by using the set of all test inputs as query set as originally suggested in Tresp (2000) would be impractical in this experiment because the BCM requires inverting a covariance matrix of the size of the query set which, considering the numbers of test inputs in this experiment, would be computationally intractable. To circumvent this problem we use the BCM algorithm to query one test input at a time. This approach is in-line with that adopted by Deisenroth and Ng (2015), where the authors did not address determining joint predictive distributions over multiple latent function values. For the BCM and rBCM, the number of experts is chosen so that each expert processes 200 training points. For SVIGP we use the implementation made available by the The GPy authors (2016), and we use the same configuration as in Hensman et al. (2013). As for *string GPs*, we use the symmetric sum as link function, and we run two types of experiments, one allowing for inference of change-points (String GP), and the other enforcing a single kernel configuration per input dimension (String GP*). The parameters $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are

[8] For instance the predictive distribution of the value of the latent function at a test input x_1, namely $f(x_1)$, obtained by using $\{x_1\}$ as set of test inputs in the BCM, differs from the predictive distribution obtained by using $\{x_1, x_2\}$ as set of test inputs in the BCM and then marginalising with respect to the second input x_2.

chosen so that the prior mean number of change-points in each input dimension be 5% of the number of distinct training and testing values in that the input dimension, and so that the prior variance of the foregoing number of change-points be 50 times the prior mean, so as to be uninformative about the number of change-points. We run $10,000$ iterations of our RJ-MMCMC sampler and discarded the first $5,000$ as 'burn-in'. After burning we record the states of the Markov chains for analysis using a 1-in-100 thinning rate. Predictive accuracies are reported in Table 5.3 and CPU time requirements[9] are illustrated in Figure 5.7.

The BCM and the rBCM perform the worst in this experiment both in terms of predictive accuracy (Table 5.3) and total CPU time (Figure 5.7). The poor scalability of the BCM and the rBCM is primarily due to the testing phase. Indeed, if we denote M the total number of experts, then $M = \lceil \frac{N}{300} \rceil$, as each expert processes 200 training points, of which there are $\frac{2}{3}N$. In the prediction phase, each expert is required to make predictions about all $\frac{1}{3}N$, which requires evaluating M products of an $\frac{1}{3}N \times 200$ matrix with a 200×200 matrix, which results in a total CPU time requirement that grows in $\mathcal{O}(M\frac{1}{3}N200^2)$, which is the same as $\mathcal{O}(N^2)$. Given that training CPU time grows linearly in N the total CPU time grows quadratically in N. This is well illustrated in Figure 5.7, where it can be seen that the slopes of total CPU time profiles of the BCM and the rBCM in log-log scale are approximately 2. The airline delays dataset was also considered by Deisenroth and Ng (2015), but the authors restricted themselves to a fixed size of the test set of $100,000$ points. However, this limitation might be restrictive as in many 'smoothing' applications, the test dataset can be as large as the training dataset—neither the BCM nor the rBCM would be sufficiently scalable in such applications.

As for SVIGP, although it was slightly more accurate than *string GPs* on this dataset, it can be noted from Figure 5.7 that *string GPs* required 10 times less CPU resources. In fact we were unable to run the experiment on the full dataset with SVIGP—we gave up after 500 CPU hours, or more than a couple of weeks wall-clock time given that the GPy implementation of SVIGP makes little use of multiple cores. As a comparison, the full experiment took 91.0 hours total CPU time ($\approx$ 15 hours wall-clock time an 8 cores i7 machine) when change-points were inferred and 83.11 hours total CPU time ($\approx$ 14 hours wall-clock time on an 8 cores i7 machine). Another advantage of additively separable *string GPs* over GPs, and subsequently over SVIGP, is that they are more interpretable. Indeed, one can determine at a glance from the learned posterior mean *string GPs* of Figure 5.8 the effect of each of the 8 covariates

[9]Defined as the total CPU clock resource usage of the whole experiment (training and testing)

Table 5.3 Predictive mean squared errors (MSEs) ± one standard error on the airline arrival delays experiment. Squared errors were expressed as fraction of the sample variance of airline arrival delays so that an MSE of 1.00 is as good as using the training mean arrival delays as predictor. The * in String GP* indicates that inference was performed without allowing for change-points. N/A entries correspond to experiments that were not over after 500 CPU hours.

N	String GP	String GP*	BCM	rBCM	SVIGP
$10,000$	1.03 ± 0.10	1.06 ± 0.10	1.06 ± 0.10	1.06 ± 0.10	$\mathbf{0.90 \pm 0.09}$
$100,000$	0.93 ± 0.03	0.96 ± 0.03	1.66 ± 0.03	1.04 ± 0.04	$\mathbf{0.88 \pm 0.03}$
$1,000,000$	0.93 ± 0.01	0.92 ± 0.01	N/A	N/A	$\mathbf{0.82 \pm 0.01}$
$5,929,413$	0.90 ± 0.01	0.93 ± 0.01	N/A	N/A	N/A

considered on arrival delays. It turns out that the three most informative factors in predicting arrival delays are departure time, distance and arrival time, while the age of the aircraft, the day of the week and the day of the month seem to have little to no effect. Finally, posterior distributions of the number of change-points are illustrated in Figure 5.9, and posterior distributions of the locations of change-points are illustrated in Figure 5.10.

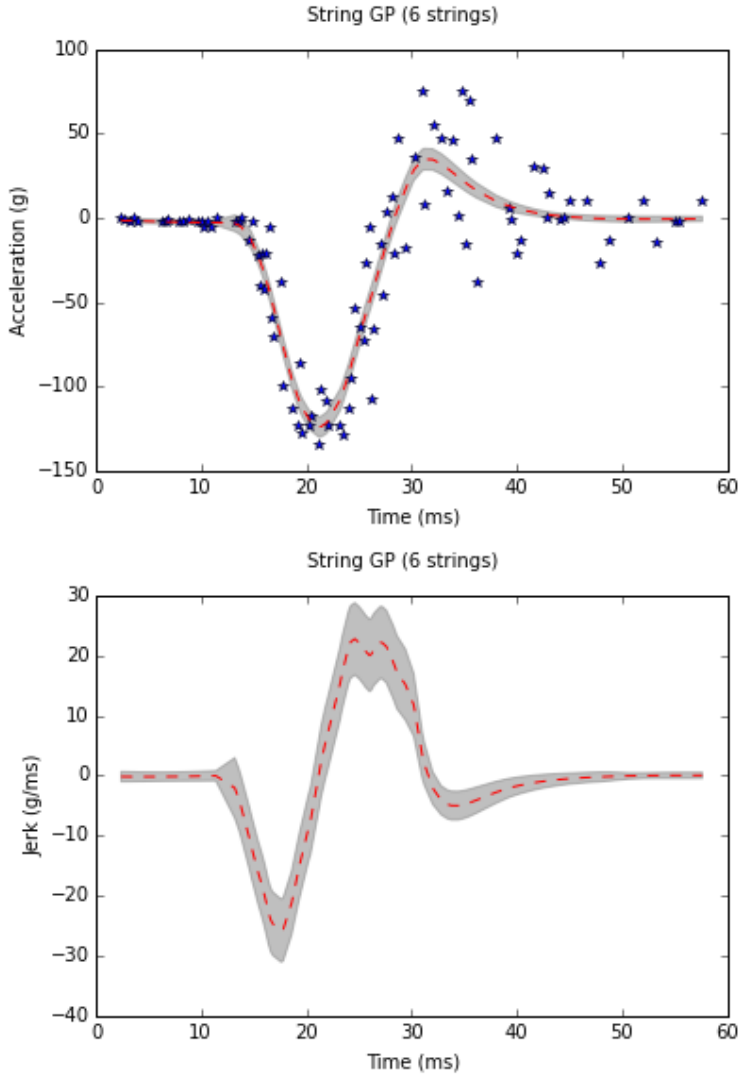

Fig. 5.5 Posterior mean ± 2 predictive standard deviations on the motorcycle dataset (see Silverman, 1985), under a Matern 3/2 derivative *string GP* prior with 6 learned strings. The top figure shows the noisy accelerations measurements and the learned latent function. The bottom function illustrates the derivative of the acceleration with respect to time learned from noisy acceleration samples. Posterior confidence bands are over the latent functions rather than noisy measurements, and as such they do not include the measurement noise.

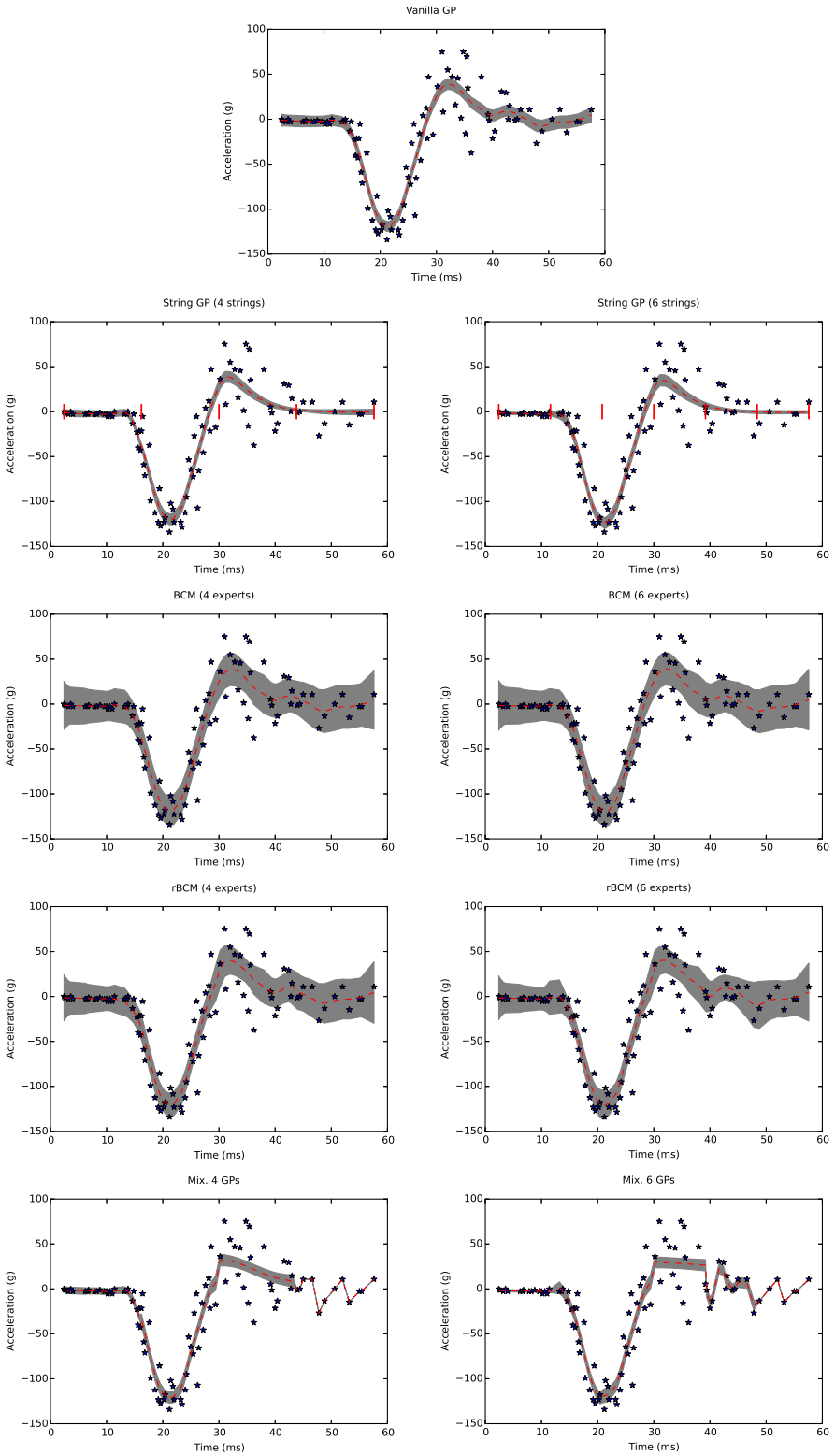

Fig. 5.6 Bayesian nonparametric regressions on the motorcycle dataset of Silverman (1985). Models compared are *string GP* regression, vanilla GP regression, mixture of independent GP regression experts on a partition of the domain, the Bayesian committee machine (BCM) and the robust Bayesian committee machine (rBCM). Domain partitions were learned during *string GP* maximum likelihood inference (red vertical bars), and reused in other experiments. Blue stars are noisy samples, red lines are posterior means of the latent function and grey bands correspond to $\pm$ 2 predictive standard deviations of the (noise-free) latent function about its posterior mean.

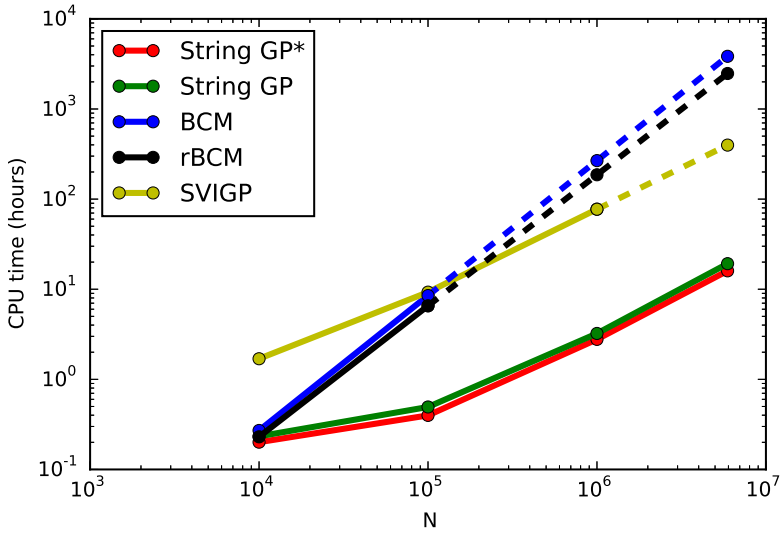

Fig. 5.7 Total CPU time (training and testing) taken by various regression approaches on the airline delays dataset as a function of the size of the subset considered. The experimental setup is described in Section 5.5.3. The plot is in log-log scale. The CPU time reflects actual CPU clock resource usage in each experiment, and is therefore agnostic to the number of CPU cores used. It can be regarded as the wall-clock time the experiment would have taken to complete on a single-core computer (with the same CPU frequency). Dashed lines are extrapolated values, and correspond to experiments that did not complete after 500 hours CPU time.

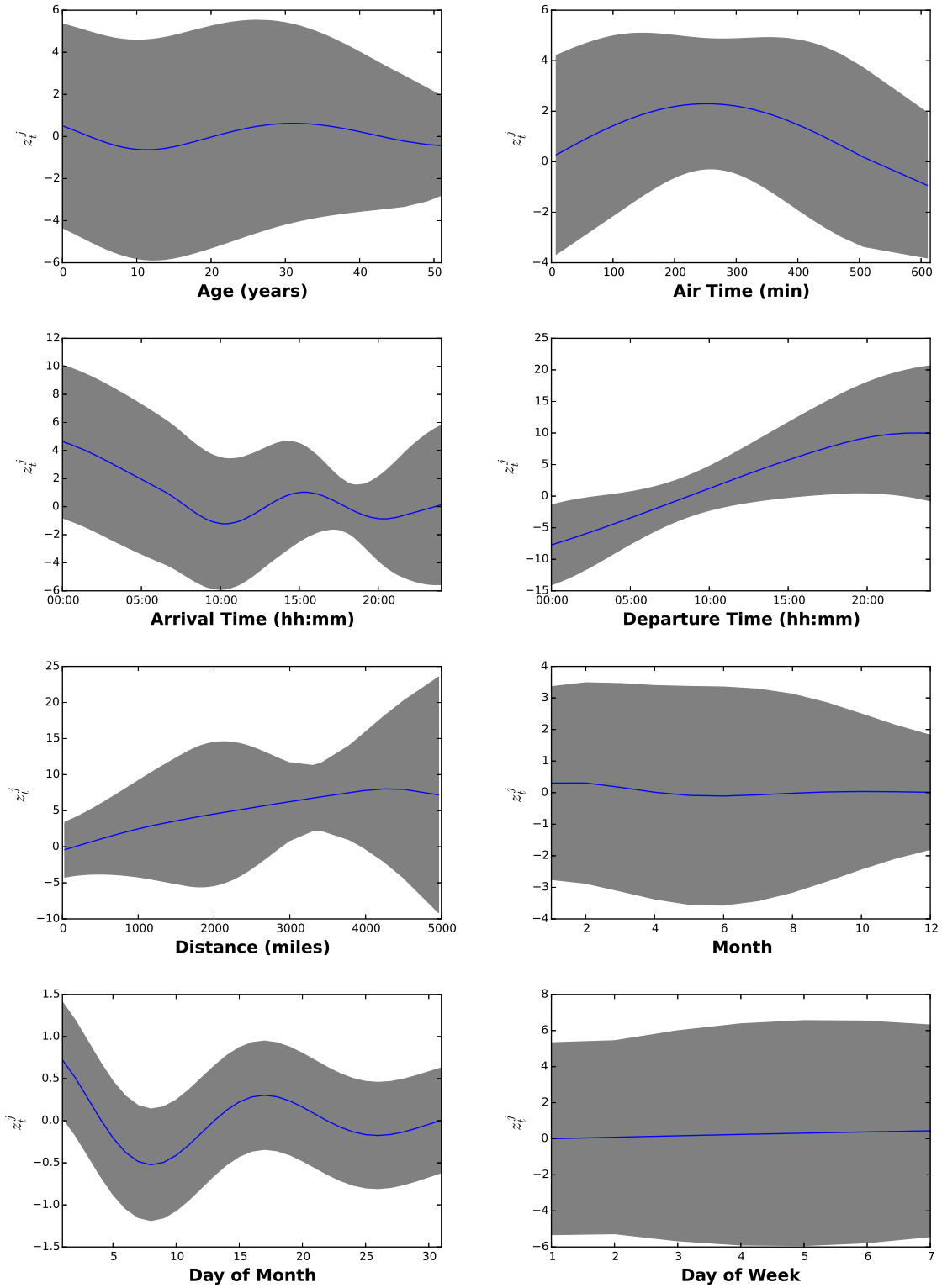

Fig. 5.8 Posterior mean ± one posterior standard deviation of univariate *string GPs* in the airline delays experiment of Section 5.5.3. Change-points were automatically inferred in this experiment.

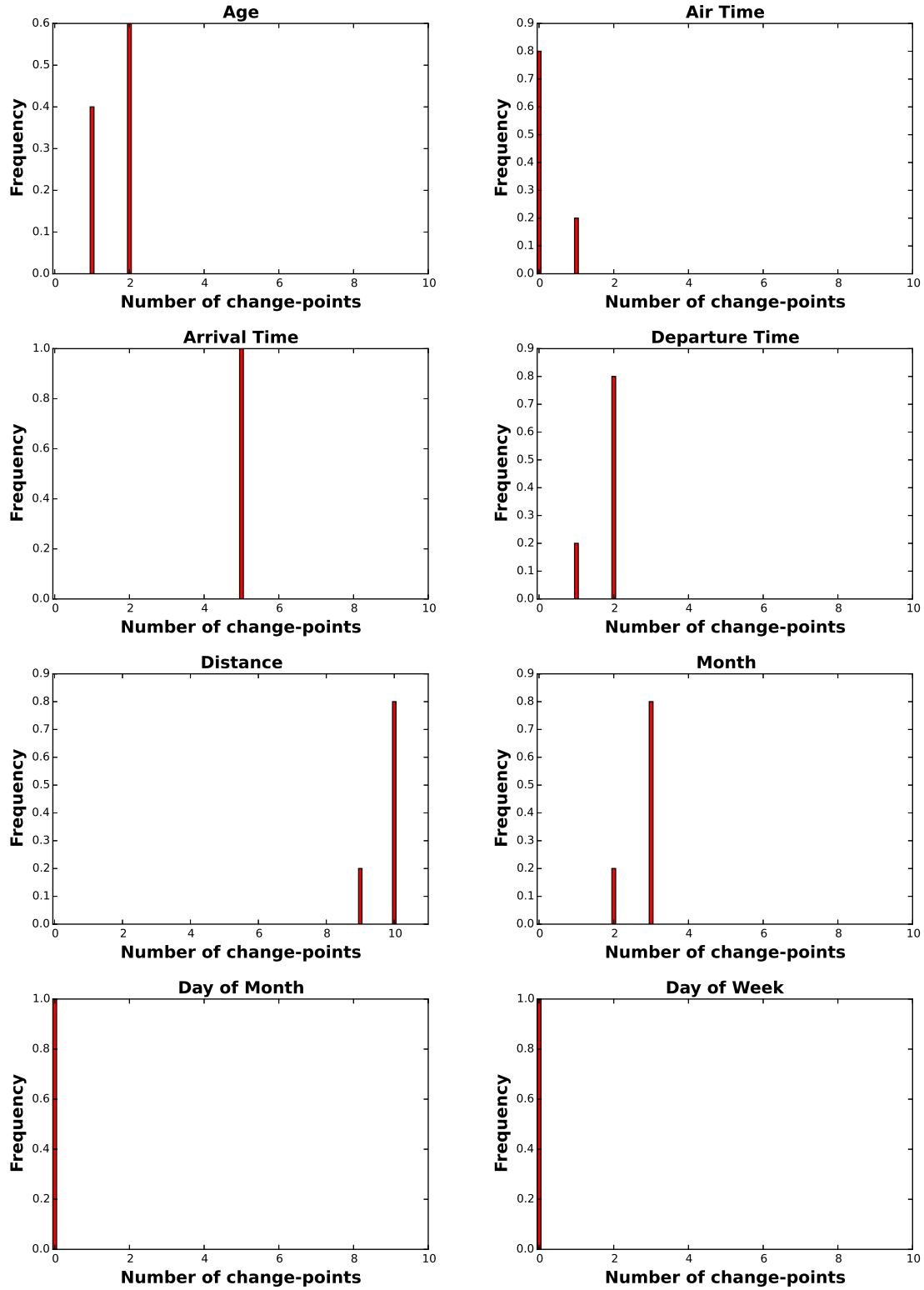

Fig. 5.9 Posterior distributions of the numbers of change-points in each input dimension in the airline delays experiment of Section 5.5.3.

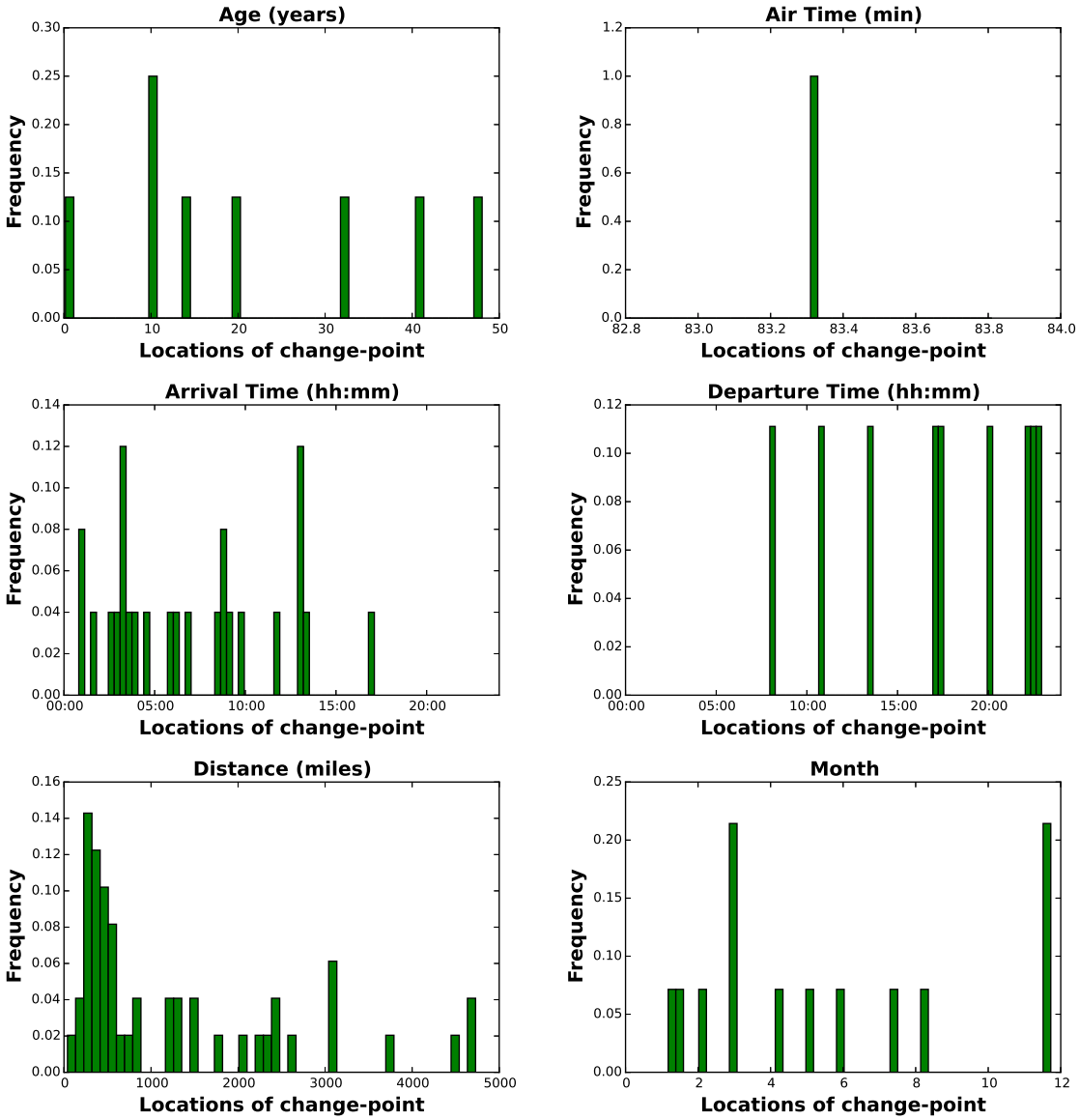

Fig. 5.10 Posterior distributions of the locations of change-points in each input dimension in the airline delays experiment of Section 5.5.3.

5.5.4 Large Scale Dynamic Portfolio Allocation

An important feature of our proposed RJ-MCMC sampler (Algorithm 5.1) is that it is agnostic with regard to the likelihood model, so long as it takes the form $p(\mathcal{D}|\mathbf{f}, \mathbf{u})$. Thus, it may be used as is on a wide variety of problems that go beyond celebrated examples such as Bayesian nonparametric regression and classification. This experiment aims at illustrating one such problem.

Background

Let $(x_i(t))_{t>0}$ for $i = 1, \ldots, n$ be n stock price processes. Let $(X_i(t))_{t>0}$ for $i = 1, \ldots, n$ denote the *market capitalisation* processes, that is $X_i(t) = n_i(t)x_i(t)$ where $n_i(t)$ is the number of shares in company i trading in the market at time t. We call *long-only portfolio* any vector-valued stochastic process $\pi = (\pi_1, \ldots, \pi_n)$ taking value on the unit simplex on $\mathbb{R}^n$, that is

$$\forall i, t, \;\; \pi_i(t) \geq 0 \quad \text{and} \quad \sum_{i=1}^{n} \pi_i(t) = 1.$$

Each process π_i represents the proportion of an investor's wealth invested in (holding) shares in asset i. An example long-only portfolio is the market portfolio $\mu = (\mu_1, \ldots, \mu_n)$ where

$$\mu_i(t) = \frac{X_i(t)}{X_1(t) + \cdots + X_n(t)} \tag{5.34}$$

is the market weight of company i at time t, that is its size relative to the total market size (or that of the universe of stocks considered). The market portfolio is very important to practitioners as it is often perceived not be to subject to idiosyncraticities, but only to systemic risk. It is often used as an indicator of how the stock market (or a specific universe of stocks) performs as a whole. We denote Z_π the value process of a portfolio π with initial capital $Z_\pi(0)$. That is, $Z_\pi(t)$ is the wealth at time t of an investor who had an initial wealth of $Z_\pi(0)$, and dynamically re-allocated all his wealth between the n stocks in our universe up to time t following the continuous-time strategy π.

A mathematical theory has recently emerged, namely *stochastic portfolio theory* (SPT) (see Karatzas and Fernholz, 2009) that studies the stochastic properties of the wealth processes of certain portfolios called *functionally-generated portfolio* under realistic assumptions on the market capitalisation processes $(X_i(t))_{t>0}$. Functionally-generated portfolios are rather specific in that the allocation at time t, namely $(\pi_1(t), \ldots, \pi_n(t))$, solely depends on the market weights vector $(\mu_1(t), \ldots, \mu_n(t))$.

Nonetheless, some functionally-generated portfolios π^* have been found that, under the mild (so-called diversity) condition

$$\exists \mu_{\max}, 0 < \mu_{\max} < 1 \quad \text{s.t.} \quad \forall i \leq n,\ t \leq T, \quad \mu_i(t) \leq \mu_{\max}, \tag{5.35}$$

outperform the market portfolio over the time horizon $[0, T]$ with probability one (see Karatzas and Fernholz, 2009; Vervuurt and Karatzas, 2015). More precisely,

$$\mathbb{P}\left(Z_{\pi^*}(T) \geq Z_\mu(T)\right) = 1 \quad \text{and} \quad \mathbb{P}\left(Z_{\pi^*}(T) > Z_\mu(T)\right) > 0. \tag{5.36}$$

Galvanized by this result, we herein consider the inverse problem consisting of learning from historical market data a portfolio whose wealth process has desirable user-specified properties. This inverse problem is perhaps more akin to the problems faced by investment professionals: i) their benchmarks depend on the investment vehicles pitched to investors and may vary from one vehicle to another, ii) they have to take into account liquidity costs, and iii) they often find it more valuable to go beyond market weights and leverage multiple company characteristics in their investment strategies.

Model Construction

We consider portfolios $\pi^f = \left(\pi_1^f, \ldots, \pi_n^f\right)$ of the form

$$\pi_i^f(t) = \frac{f(c_i(t))}{f(c_1(t)) + \cdots + f(c_n(t))}, \tag{5.37}$$

where $c_i(t) \in \mathbb{R}^d$ are some quantifiable characteristics of asset i that may be observed in the market at time t, and f is a positive-valued function. Portfolios of this form include all functionally-generated portfolios studied in SPT as a special case.[10] A crucial departure of our approach from the aforementioned type of portfolios is that the market characteristics processes c_i need not be restricted to size-based information, and may contain additional information such as social media sentiments, stock price path-properties, but also characteristics relative to other stocks such as performance relative to the best/worst performing stock last week/month/year etc. We place a mean-zero *string GP* prior on $\log f$. Given some historical data $\mathcal{D}$ corresponding to a training time horizon $[0, T]$, the likelihood model $p\left(\mathcal{D}|\pi^f\right)$ is defined by the investment professional and reflects the extent to which applying the investment strategy π^f over

[10] We refer the reader to Karatzas and Fernholz (2009) for the definition of functionally-generated portfolios.

the training time horizon would have achieved a specific investment objective. An example investment objective is to achieve a high excess return relative to a benchmark portfolio α

$$\mathcal{U}_{\text{ER}}\left(\pi^f\right) = \log Z_{\pi^f}(T) - \log Z_\alpha(T). \tag{5.38}$$

Here α can be the market portfolio (as in SPT) or any stock index. Other risk-adjusted investment objectives may also be used. One such objective is to achieve a high Sharpe-ratio, defined as

$$\mathcal{U}_{\text{SR}}\left(\pi^f\right) = \frac{\bar{r}\sqrt{252}}{\sqrt{\frac{1}{T}\sum_{t=1}^{T}(r(t)-\bar{r})^2}}, \tag{5.39}$$

where the time t is in days, $r(t) := \log Z_{\pi^f}(t) - \log Z_{\pi^f}(t-1)$ are the daily returns the portfolio π^f and $\bar{r} = \frac{1}{T}\sum_{t=1}^{T} r(t)$ its average daily return. More generally, denoting $\mathcal{U}\left(\pi^f\right)$ the performance of the portfolio π^f over the training horizon $[0, T]$ (as per the user-defined investment objective), we may choose as likelihood model a distribution over $\mathcal{U}\left(\pi^f\right)$ that reflects what the investment professional considers good and bad performance. For instance, in the case of the excess return relative to a benchmark portfolio or the Sharpe ratio, we may choose $\mathcal{U}\left(\pi^f\right)$ to be supported on $]0, +\infty[$ (for instance $\mathcal{U}\left(\pi^f\right)$ can be chosen to be Gamma distributed) so as to express that portfolios that do not outperform the benchmark or loose money overall in the training data are not of interest. We may then choose the mean and standard deviation of the Gamma distribution based on our expectation as to what performance a good candidate portfolio can achieve, and how confident we feel about this expectation. Overall we have,

$$p\left(\mathcal{D}|\pi^f\right) = \gamma\left(\mathcal{U}\left(\pi^f\right); \alpha_e, \beta_e\right), \tag{5.40}$$

where $\gamma(.; \alpha, \beta)$ is the probability density function of the Gamma distribution. Noting, from Equation (5.37) that $\pi^f(t)$ only depends on f through its values at $(c_1(t), \ldots, c_n(t))$, and assuming that $\mathcal{U}\left(\pi^f\right)$ only depends on π^f evaluated at a finite number of times (as it is the case for excess returns and the Sharpe ratio), it follows that $\mathcal{U}(\pi^f)$ only depends on $\mathbf{f}$, a vector of values of f at a finite number of points. Hence the likelihood model, which we may rewrite as

$$p(\mathcal{D}|\mathbf{f}) = \gamma\left(\mathcal{U}\left(\pi_i^f\right); \alpha_e, \beta_e\right), \tag{5.41}$$

is of the form required by the RJ-MCMC sampler previously developed. By sampling from the posterior distribution $p(\mathbf{f}, \mathbf{f}^*, \nabla\mathbf{f}, \nabla\mathbf{f}^*|\mathcal{D})$, the hope is to learn a portfolio that

did well during the training horizon, to analyse the sensitivity of its investment strategy to the underlying market characteristics through the gradient of f, and to evaluate the learned investment policy on future market conditions.

Experimental Setup

The universe of stocks we considered for this experiment are the constituents of the S&P 500 index, accounting for changes in constituents over time and corporate events. We used the period 1st January 1990 to 31st December 2004 for training and we tested the learned portfolio during the period 1st January 2005 to 31st December 2014. We rebalanced the portfolio daily, giving a total of 2.52 million input points at which the latent function f must be learned. We considered as market characteristics the market weight (CAP), the latest return on asset (ROA) defined as the ratio between the net yearly income and the total assets as per the latest balance sheet of the company known at the time of investment, the previous close-to-close return (PR), the close-to-close return before the previous (PR2), and the S&P long and short term credit rating (LCR and SCR). While the market weight is a company size characteristic, the ROA reflects how well a company performs relative to its size, and we hope that S&P credit ratings will help further discriminate successful companies from others. The close-to-close returns are used to learn possible 'momentum' patterns from the data. The data originate from the CRSP and Compustat databases. In the experiments we considered as performance metric the annualised excess return $\mathcal{U}_{\text{ER-EWP}}$ relative to the equally-weighted portfolio. We found the equally-weighted portfolio to be a harder benchmark to outperform than the market portfolio. We chose α_e and β_e in Equation (5.41) so that the mean of the Gamma distribution is 10.0 and its variance 0.5, which expresses a very greedy investment target.

It is worth pointing out that none of the scalable GP alternatives previously considered can cope with our likelihood model Equation (5.41). We compared the performance of the learned *string GP* portfolio out-of-sample to those of the best three SPT portfolios studied in Vervuurt and Karatzas (2015), namely the equally weighted portfolio

$$\pi_i^{\text{EWP}}(t) = \frac{1}{n}, \tag{5.42}$$

and the diversity weighted portfolios

$$\pi_i^{\text{DWP}}(t;p) = \frac{\mu_i(t)^p}{\mu_1(t)^p + \cdots + \mu_n(t)^p}, \tag{5.43}$$

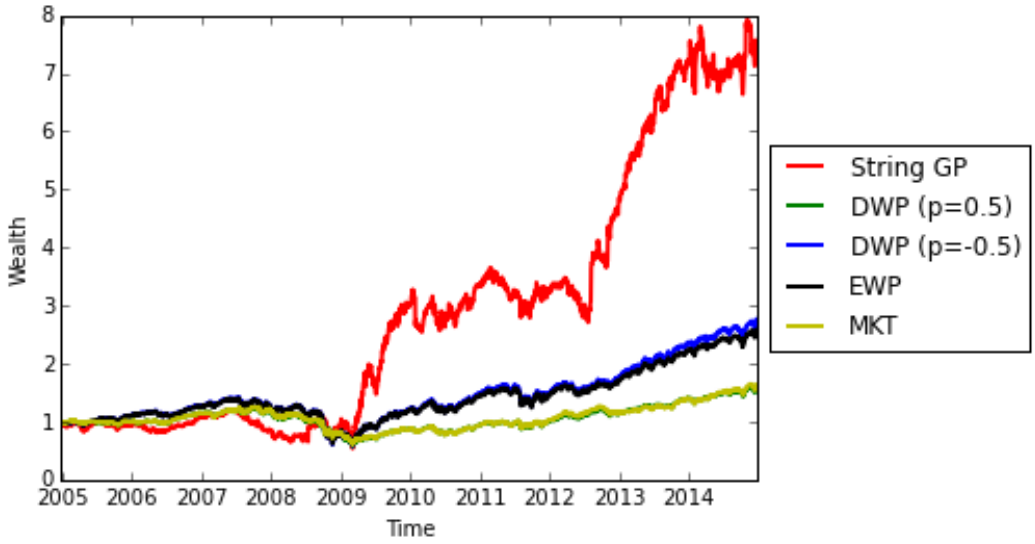

Fig. 5.11 Evolution of the wealth processes of various long-only trading strategies on the S&P 500 universe of stocks between 1st January 2005 (where we assume a starting wealth of 1) and 31st December 2014. The String GP strategy was learned using market data from 1st January 1990 to 31st December 2004 as descried in Section 5.5.4. EWP refers to the equally-weighted porfolio, MKT refers to the market portfolio (which weights stocks proportionally to their market capitalisations) and DWP (p) refers to the diversity-weighted portfolio with exponent p (which weights stocks proportionally to the p-th power of their market weights).

with parameter p equals to -0.5 and 0.5, and the market portfolio. Results are provided Table 5.4, and Figure 5.11 displays the evolution of the wealth process of each strategy. It can be seen that the learned *string GP* strategy considerably outperforms the next best SPT portfolio. This experiment not only demonstrates that *string GPs* scale to large scale problems, it also illustrates that our inference scheme is able to unlock commercial value in new intricate large scale applications where no alternative is readily available.

Table 5.4 Performance of various long-only trading strategies on the S&P 500 universe of stocks between 1st January 2005 (where we assume a starting wealth of 1) and 31st December 2014. The String GP strategy was learned using market data from 1st January 1990 to 31st December 2004 as descried in Section 5.5.4. EWP refers to the equally-weighted porfolio, MKT refers to the market portfolio (which weights stocks proportionally to their market capitalisations) and DWP (p) refers to the diversity-weighted portfolio with exponent p (which weights stocks proportionally to the p-th power of the market weight of the asset). $Z_\pi(T)$ denotes the terminal wealth of strategy π, and Avg. Ann. Ret. is the strategy's equivalent constant annual return over the test horizon.

Strategy	Sharpe Ratio	$Z_\pi(T)/Z_{\text{EWP}}(T)$	Avg. Ann. Ret.
String GP	**0.73**	**2.87**	**22.07%**
DWP ($p = -0.5$)	0.55	1.07	10.56%
EWP	0.53	1.00	9.84%
MKT	0.34	0.62	4.77%
DWP ($p = 0.5$)	0.33	0.61	4.51%

5.6 Discussion

5.6.1 Limitations

The main limitation of our approach is that, unlike the *standard GP paradigm* in which the time complexity of marginal likelihood evaluation does not depend on the dimension of the input space, the *string GP paradigm* requires a number of computing cores that increases linearly with the dimension of the input space, or alternatively has a time complexity linear in the input space dimension on single-core machines. This is a by-product of the fact that in the *string GP paradigm*, we jointly infer the latent function and its gradient. If the gradient of the latent function is inferred in the *standard GP paradigm*, the resulting complexity will also be linear in the input dimension. That being said, overall our RJ-MCMC inference scheme will typically scale better per iteration to large input dimensions than gradient-based marginal likelihood inference in the *standard GP paradigm*, as the latter typically requires numerically evaluating an Hessian matrix, which requires computing the marginal likelihood a number of times per iterative update that grows quadratically with the input dimension. In contrast, a Gibbs cycle in our MCMC sampler has worst case time complexity that is linear in the input dimension.

5.6.2 Extensions

Distributed String GPs

Firstly, the RJ-MCMC inference scheme we propose may be easily adapted to handle applications where the dataset is so big that it has to be stored across multiple clusters, and inference techniques have to be developed as data flow graphs[11] (for instance using libraries such as TensorFlow).

To do so, the choice of string boundary times can be adapted so that each string has the same number of inner input coordinates, and such that in total there are as many strings across dimensions as a target number of available computing cores. We may then place a prior on kernel memberships similar to that of previous sections. Here, the change-points may be restricted to coincide with boundary times, and we may choose priors such that the sets of change-points are independent between input dimensions. In each input dimension the prior on the number of change-points can be chosen to be a truncated Poisson distribution (truncated to never exceed the total number of boundary times), and conditional on their number we may choose change-points to be uniformly distributed in the set of boundary times. In so doing, any two strings whose shared boundary time is not a change-point will be driven by the same kernel configuration.

This new setup presents no additional theoretical or practical challenges, and the RJ-MCMC techniques previously developed are easily adaptable to jointly learn change-points and function values. Unlike the case we developed in previous section where an update of the univariate *string GP* corresponding to an input dimension, say the j-th, requires looping through all distinct j-th input coordinates, here no step in the inference scheme requires a full view of the dataset in any input dimension. Full RJ-MCMC inference can be constructed as a data flow graph. An example such graph is constructed as follows. The leaves correspond to computing cores responsible for generating change-points and kernel configurations, and mapping strings to kernel configurations. The following layer is made of compute cores that use kernel configurations coming out of the previous layer to sequentially compute boundary conditions corresponding to a specific input dimension—there are d such compute cores, where d is the input dimension. These compute cores then pass computed boundary conditions to subsequent compute cores we refer to as string compute cores. Each string compute core is tasked with computing *derivative string GP* values for a specific input dimension and for a specific string in

[11] A data flow graph is a computational (directed) graph whose nodes represent calculations (possibly taking place on different computing units) and directed edges correspond to data flowing between calcultions or computing units.

that input dimension, conditional on previously computed boundary conditions. These values are then passed to a fourth layer of compute cores, each of which being tasked with computing function and gradient values corresponding to a small subset of training inputs from previously computed *derivative string GP* values. The final layer then computes the log-likelihood using a distributed algorithm such as *MapReduce* (Dean and Ghemawat (2008)) when possible. This proposal data flow graph is illustrated Figure 5.12.

We note that the approaches of Kim et al. (2005), Gramacy and Lee (2008), Tresp (2000), and Deisenroth and Ng (2015) also allow for fully-distributed inference on regression problems. Distributed *string GP* RJ-MCMC inference improves on these in that it places little restriction on the type of likelihood. Moreover, unlike Kim et al. (2005) and Gramacy and Lee (2008) that yield discontinuous latent functions, *string GPs* are continuously differentiable, and unlike Tresp (2000) and Deisenroth and Ng (2015), local experts in the *string GP paradigm* (i.e. strings) are driven by possibly different sets of hyper-parameters, which facilitates the learning of local patterns.

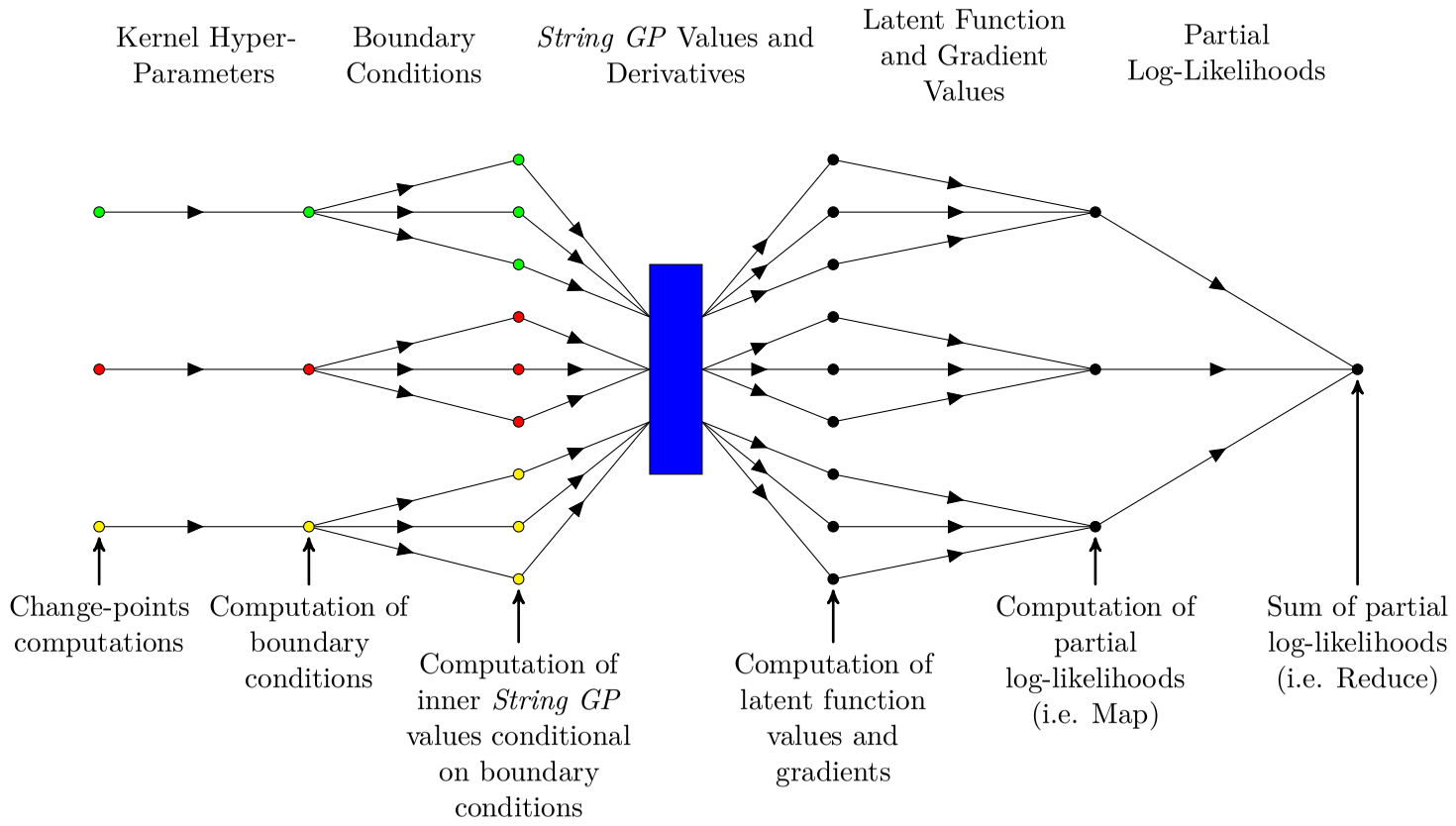

Fig. 5.12 Example data flow graph for fully-distributed *string GP* inference under an i.i.d observations likelihood model. Here the input space is three-dimensional to ease illustration. Filled circles represent compute cores, and edges correspond to flows of data. Compute cores with the same colour (green, red or yellow) perform operations pertaining to the same input dimension, while black-filled circles represent compute cores performing cross-dimensional operations. The blue rectangle plays the role of a hub that relays *string GP* values to the compute cores that need them to compute the subset of latent function values and gradients they are responsible for. These values are then used to compute the log-likelihood in a distributed fashion using the *MapReduce* algorithm. Each calculation in the corresponding RJ-MCMC sampler would be initiated at one of the compute cores, and would trigger updates of all edges accessible from that compute core.

Approximate MCMC for i.i.d. Observations Likelihoods

As discussed in Section 5.2, the bottleneck of our proposed inference scheme is the evaluation of the likelihood. When the likelihood factorises across training samples, the linear time complexity of our proposed approach can be further reduced using a Monte Carlo approximation of the log-likelihood (see for instance Bardenet et al. (2014) and references therein). Although the resulting Markov chain will typically not converge to the true posterior distribution, in practice its stationary distribution can be sufficiently close to the true posterior when reasonable Monte Carlo sample sizes are used in the likelihood approximation. Convergence results of such approximations have recently been studied by Bardenet et al. (2014) and Alquier et al. (2016). We expect this extension to speed-up inference when the number of compute cores is in the order of magnitude of the input dimension, but we would recommend the previously mentioned fully-distributed string GP inference extension when compute cores are not scarce.

Variational Inference

It would be useful to develop suitable variational methods for inference under *string GP* priors, that we hope will scale similarly to our proposed RJ-MCMC sampler but will converge faster. We anticipate that the main challenge here will perhaps be the learning of model complexity, that is the number of distinct kernel configurations in each input dimension.

5.7 Summary

In this chapter we propose Bayesian nonparametric inference methods using *string GPs* to learn latent functions when data might exhibit local patterns. We argue and empirically illustrate that *string GPs* present an unparalleled opportunity for learning local patterns in small scale regression problems using nothing but standard Gaussian process regression techniques. More importantly, we propose a novel *scalable* RJ-MCMC inference scheme to learn latent functions in a wide variety of machine learning tasks, while simultaneously determining *whether* the dataset exhibits local patterns, *how many* types of local patterns the data might exhibit, and *where* do changes in these patterns are likely to occur. The proposed scheme has time complexity and memory requirement that are both *linear* in the sample size N. When the number of available computing cores is at least equal to the dimension d of the input space,

the time complexity is independent from the dimension of the input space. Else, the time complexity grows as $\mathcal{O}(dN)$. The memory requirement grows as $\mathcal{O}(dN)$. We empirically illustrate that our approach scales considerably better than competing alternatives on a standard benchmark dataset, and is able to process data sizes that competing approaches cannot handle in a reasonable time.

This concludes Part II of this thesis, where we have proposed novel algorithmic solutions to various Bayesian kernel learning problems that address the need for scalability and allow the automatic learning of *whether*, *how many* and *where* local patterns might occur in datasets. In Part III, we extend our discussion to any kernel method (Bayesian or frequentist), we discuss properties a family of kernels should satisfy to be deemed 'general-purpose' and we argue that existing families of kernels are not general-purpose. We construct tractable families of kernels that are general-purpose, and we propose an algorithm for automatic, general-purpose, and scalable kernel learning.

Part III

General-Purpose and Scalable Kernel Methods

Chapter 6

Mathematical Formalism of General-Purpose Kernels

> "Nothing has such power to broaden the mind as the ability to investigate systematically and truly all that comes under thy observation in life."
>
> *Marcus Aurelius*

Part II was dedicated to providing algorithmic solutions to kernel-based Bayesian nonparametric function learning problems that are scalable and flexible enough to cope with time evolving or local patterns. In this part of the thesis we generalise our quest for scalability and flexibility to all kernel methods, Bayesian or otherwise. We begin with a discussion of the mathematical properties that families of kernels should satisfy to be theoretically guaranteed to be as flexible as required.

The choice of kernel in kernel methods is often left to the user, although it may considerably affect the performance of a machine learning task. In this chapter we review popular kernel methods and prove that, in each of them, pointwise convergence of a sequence of kernels implies pointwise convergence of the corresponding sequence of performances to that of the limit kernel. Consequently, if a family of kernels is pointwise dense in the class of continuous bounded kernels, then it contains elements that may perform as well as any oracle[1] continuous bounded kernel in most kernel methods and on the *same* dataset, up to an arbitrarily small precision.

[1] That is, one that is ideal for the task of interest.

6.1 The Limits of Universality

Notions have been proposed that aim at ascertaining the flexibility of kernels or equivalently of their RKHSs. Examples include *universality*[2] (Micchelli et al. (2006)) and (L, P)-*richness* (Steinwart and Christmann (2008); Steinwart et al. (2006)). As universality implies (L, P)-richness for loss functions of practical interest (Steinwart et al., 2006, Corollary 1), if we find universality not be a satisfactory enough notion of flexibility, neither can (L, P)-richness.

In fact, not only is universality not a satisfactory notion of flexibility for existing kernel methods, but more generally no binary notion of flexibility that is a property of a single kernel or RKHS (rather than that of a family of kernels) can be discriminative in a manner that is consistent with conventional wisdom on kernel methods. Indeed, in practice, the kernel is typically learned from a parametric family of kernels that contains an infinite (often even uncountable) number of kernels, by maximising a data-driven objective function. For the sake of the argument, let us assume that flexibility is defined as a binary property of a specific kernel or equivalently its RKHS, and let us assume that the parametric family of candidate kernels contains at least one kernel that satisfies the foregoing binary flexibility property. It then follows that either a large and possibly infinite number of kernels in the parametric family of candidate kernels (corresponding to a large set of hyper-parameters) will satisfy the binary flexibility property, in which case the notion would not be discriminative enough as it is well known that performance in most kernel methods is extremely sensitive to kernel hyper-parameters, or only a handful of kernels (corresponding to a small set of hyper-parameters) will satisfy the binary flexibility property, which would imply that there exists a small set of kernel hyper-parameters that would perform well on *all* datasets,[3] which again would be inconsistent with conventional wisdom as the appropriateness of hyper-parameters often depends on (at least the scale/units of) the training data inputs. In regards to universality specifically, for nearly every family of kernels commonly used in practice, each element in the family admits a spectral density and consequently is universal (Micchelli et al., 2006, Proposition 16).

Clearly, the hypothesis space of candidate functions expressed by any universal kernel is large enough, so where does it go wrong? A kernel method can be regarded as a function learning algorithm that takes as input a finite dataset and an RKHS

[2] We recall that a continuous kernel defined on a compact metric space $\mathcal{X}$ is said to be universal when its RKHS is pointwise dense in the space of all continuous functions defined on $\mathcal{X}$.

[3] The notion of flexibility considered here is a binary property of an RKHS and consequently is data-independent.

(playing the role of the hypothesis family of candidate functions), and returns the element in the RKHS that it considers to be the most suitable function for the problem at hand in light of the training dataset. Although it is crucial that the RKHS be large enough (for instance universal), it is even more important that the function learning algorithm explores the RKHS in a data-efficient way, which is not guaranteed by the universality property. As an illustration, let us consider a kernel method with kernel

$$k(x, y) = \exp\left(-10^{-10}||x - y||^2\right),$$

which is universal as it admits a spectral density, and let us assume that the kernel method satisfies the *representer theorem* (Schölkopf and Smola (2001)) so that the optimal solution takes the form

$$f^*(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$$

where x_i are training inputs. Example such kernel methods are numerous and include regularised Empirical Risk Minimisation and Gaussian process regression to name but a few. Given that our kernel is almost constant, despite its universality, our kernel method will explore parts of the RKHS containing functions that are not almost constant only when the size n of the training dataset is very large. However, in practice the operator often has little control over the amount of data available. That being said, other kernels may well exist that perform significantly better than k, on the *same* dataset and using the *same* kernel method (or function learning algorithm).

So how can we address these limitations? There are two options. The first option is to develop new kernel methods (or function learning algorithms) that will do a better job at uncovering good candidate functions in *any* universal kernel, irrespective of the size of the training dataset, thereby fully exploiting the universality of the family of candidate functions. This endeavour seems a bit daunting considering the number of machine learning tasks for which a kernel-based solution has been proposed. In fact, it is not even obvious that such a function learning algorithm can be obtained as a solution of an optimisation problem, or as a standard Bayesian inference problem. The second option is to develop a stronger notion of flexibility that is consistent with conventional wisdom on kernel methods, and that provides performance guarantees in most existing kernel methods. This is the approach we will be adopting in the rest of this chapter. As previously discussed, a needed departure from universality is that

such a notion of flexibility has to be a property of a family of kernels rather than a single kernel.

6.2 Intuitive Meaning of General-Purposeness

The Oxford English Dictionary defines 'general-purpose' as 'having a range of potential uses or functions'. 'General-purpose' families of kernels should therefore be families of kernels that are as useful as, and may be used as alternatives to any other family of kernels. In practice, one is often interested in working with hypothesis spaces of functions that are at least continuous. This requires the reproducing kernel of the hypothesis RKHS (or the covariance function in a Bayesian nonparametric setting) to be continuous. Moreover, we may consider the kernel to be bounded without loss of generality. Indeed, unless one works with a nonstationary kernel on an input space that is required to be unbounded, the kernel will effectively be bounded.[4] Thus, for a family of kernels $\mathcal{K}_\Theta = \{k_\theta\}$ to be general-purpose, it has to be the case that for any oracle continuous bounded kernel k, an element $k_\theta \in \mathcal{K}_\Theta$ should exist that 'performs as well as' k on the task of interest, within an arbitrarily small precision. What it takes for this assertion to hold true inherently depends on the meaning of 'performing as well as', which may vary from one machine learning task to another. However, we will review a few popular kernel methods and the dependencies of their solutions to the kernel used, say k, to argue that, for most kernel methods, a sufficient condition for the above assertion to hold true is that the family $\mathcal{K}_\Theta$ be dense in the family of all continuous bounded kernels with respect to the pointwise convergence of functions.[5]

6.3 General-Purpose Kernels in Gaussian Process Regression

GP regression (Rasmussen and Williams (2005)) with covariance function k, is tantamount to evaluating a predictive distribution $p(\mathbf{f}^*|\mathbf{f}, \mathbf{y}, k)$ that is Gaussian with mean

$$\mathbf{K}_{\mathbf{x}^*,\mathbf{x}} \left(\mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{y} \tag{6.1}$$

[4] All stationary kernels h are bounded as $\forall \tau, |h(\tau)| \leq h(0)$, and continuous nonstationary kernels on a bounded domain are bounded like any continuous function on a bounded domain.

[5] That is for every continuous bounded kernel k, there exists a sequence $k_{\theta_i} \in \mathcal{K}_\Theta$ such that $\forall u, v,\ k(u,v) = \lim_{i \to +\infty} k_{\theta_i}(u,v)$.

where $\mathbf{I}$ is the identity matrix, and covariance matrix

$$\mathbf{K}_{\mathbf{x}^*,\mathbf{x}^*} - \mathbf{K}_{\mathbf{x}^*,\mathbf{x}} \left(\mathbf{K}_{\mathbf{x},\mathbf{x}} + \sigma^2 \mathbf{I}\right)^{-1} \mathbf{K}_{\mathbf{x}^*,\mathbf{x}}^T, \tag{6.2}$$

where $\mathbf{K}_{\mathbf{x},\mathbf{x}} = [k(x_i, x_j)]_{i,j}$, $\mathbf{K}_{\mathbf{x}^*,\mathbf{x}^*} = [k(x_i^*, x_j^*)]_{i,j}$ and $\mathbf{K}_{\mathbf{x}^*,\mathbf{x}} = [k(x_i^*, x_j)]_{i,j}$. As this Gaussian predictive probability density function depends on the kernel k only as a continuous function of the matrices $\mathbf{K}_{\mathbf{x},\mathbf{x}}$, $\mathbf{K}_{\mathbf{x}^*,\mathbf{x}}$, and $\mathbf{K}_{\mathbf{x}^*,\mathbf{x}^*}$, if there exists a sequence of kernels k_{θ_i} that converges to k pointwise, then the sequence of conditional (predictive) random variables $\mathbf{f}^*|\mathbf{f}, \mathbf{y}, k_{\theta_i}$ converges to $\mathbf{f}^*|\mathbf{f}, \mathbf{y}, k$ in distribution. In particular, the sequence of predictive means resulting from using the kernels k_{θ_i} converges to the predictive mean corresponding to the kernel k. Hence, when the predictive mean Equation (6.1) is used as point estimate of the values of the latent function as it is often the case in practice, the corresponding sequences of mean squared errors (MSEs) and mean absolute errors (MAEs) converge respectively to the mean square error and mean absolute error obtained with the kernel k. Thus, whether performance is thought of as properties of the predictive distribution, mean square errors or mean absolute errors, pointwise convergence of a sequence of kernels k_{θ_i} to k implies convergence of the corresponding sequence of performances to that of k, and consequently we may always find a kernel k_{θ_N} in the family that performs as well as k up to an arbitrarily small precision.

6.4 General-Purpose Kernels in other Gaussian Process Models

More generally, for any Bayesian nonparametric inference of a latent function f under a GP prior with covariance function k, when the likelihood model $p(\mathcal{D}|\mathbf{f})$ is a continuous and bounded function of $\mathbf{f}$, the values of the latent function at training inputs, as it is often the case in practice (e.g. GP classification), pointwise convergence of k_{θ_i} to k implies convergence of the predictive densities $p(\mathbf{f}^*|\mathcal{D}, k_{\theta_i})$ to $p(\mathbf{f}^*|\mathcal{D}, k)$, where $\mathbf{f}^*$ denotes the values of the latent function at test inputs, and more importantly convergence of the model evidences $p(\mathcal{D}|k_{\theta_i})$ to $p(\mathcal{D}|k)$ (see Appendix D.1 for the proof). Note that this holds true irrespective of whether or not $p(\mathbf{f}^*|\mathcal{D}, k)$ or $p(\mathcal{D}|k)$ is analytically tractable, which makes our discussion useful to MCMC and variational methods.

6.5 General-Purpose Kernels in Regularised Empirical Risk Minimisation

Given a loss function L and a kernel k, a regularised empirical risk minimization problem over an RKHS $\mathcal{H}_k$ with reproducing kernel k

$$\min_{f \in \mathcal{H}_k} \quad \frac{1}{n} \sum_{i=1}^{n} L(y_i, f(x_i)) + \lambda \left\| f \right\|_{\mathcal{H}_k}^2 \tag{6.3}$$

has by the *representer theorem* (Schölkopf and Smola (2001)) a solution of form $f_k^*(x) = \sum_{i=1}^n \alpha_i k(x, x_i)$, so that it can be rewritten as

$$\min_{\boldsymbol{\alpha} \in \mathbb{R}^n} \quad \mathcal{L}\left(\boldsymbol{\alpha}, \mathbf{K}_{\mathbf{x},\mathbf{x}}\right),$$

with

$$\mathcal{L}\left(\boldsymbol{\alpha}, \mathbf{K}_{\mathbf{x},\mathbf{x}}\right) = \frac{1}{n} \sum_{i=1}^{n} L\left(y_i, \mathbf{K}_{x_i,\mathbf{x}} \boldsymbol{\alpha}\right) + \lambda \boldsymbol{\alpha}^\top \mathbf{K}_{\mathbf{x},\mathbf{x}} \boldsymbol{\alpha}.$$

By the Maximum theorem (Sundaram, 1996, Th. 9.14), if the loss function L is continuous, as it is often the case, then the optimal regularised empirical risk $\mathcal{L}\left(\boldsymbol{\alpha}^*, \mathbf{K}_{\mathbf{x},\mathbf{x}}\right)$ is a continuous function of the kernel matrix $\mathbf{K}_{\mathbf{x},\mathbf{x}}$. Therefore, pointwise convergence of a sequence of kernels k_{θ_i} to k implies convergence of the corresponding sequence of optimal regularised empirical risks to that of k. This covers many instances of kernel methods, including support vector machines (hinge loss) and kernel ridge regression (squared loss). In many cases, we may additionally obtain pointwise convergence of the family of solutions $f_{k_{\theta_i}}^*$ to f_k^*, which would imply convergence of the sequence of empirical risks of $f_{k_{\theta_i}}^*$ to that of f_k^*. This holds true for instance for kernel ridge regression, as the solutions are provably of the form of Equation (6.1), and for kernel SVM when the kernels k_{θ_i} are such that the matrices $\mathbf{K}_{\mathbf{x},\mathbf{x}}$ are all strictly positive definite (by the convex Maximum theorem (Sundaram, 1996, Th. 9.17)). Furthermore, when the solutions converge pointwise and we also have $\forall k, |L(Y, f_k^*(X))| \leq g(Y, X)$ where $g(Y, X)$ is integrable with respect to the data generating distribution (e.g. when the loss function is bounded or truncated), by dominated convergence theorem, the true risks $\mathbb{E}L\left(Y, f_{k_{\theta_i}}^*(X)\right)$ also converge to $\mathbb{E}L\left(Y, f_k^*(X)\right)$.[6]

[6] Hint of proof: by dominated convergence theorem we can exchange limit and expectation as the dominating function g is integrable. We then use the pointwise convergence of solutions and the continuity of the loss function to conclude.

6.6 General-Purpose Kernels in Kernel Principal Component Analysis

Lastly, we recall that kernel PCA (Schölkopf et al. (1997)) aims at iteratively learning a set of orthogonal functions $\{f_k^i\}$ in an RKHS $\mathcal{H}_k$ with reproducing kernel k, that maximise the empirical variance of the data that they explain:

$$f_k^i = \underset{f \perp \left\{f_k^1, \ldots, f_k^{i-1}\right\}}{\operatorname{argmax}} \mathcal{V}(f), \tag{6.4}$$

where

$$\mathcal{V}(f) = \frac{1}{n||f||_{\mathcal{H}}^2} \sum_{j=1}^{n} f(x_j)^2.$$

The solution is found to be $f_k^i(x) = \mathbf{K}_{x,\mathbf{x}}\boldsymbol{\alpha}_i$ with $\boldsymbol{\alpha}_i = \frac{1}{\sqrt{\lambda_i}} e_i$, where $(e_1, \ldots, e_n)$ are orthonormal eigenvectors of $\mathbf{K}_{\mathbf{x},\mathbf{x}}$ with associated eigenvalues $(\lambda_1, \ldots, \lambda_n)$. Moreover, the empirical variances explained by $(f_k^1, \ldots, f_k^n)$, which reflect the extent to which the RKHS $\mathcal{H}_k$, and subsequently the kernel k, accounts for the 'variability' of the data, are found to be $\mathcal{V}(f_k^i) = \frac{\lambda_i}{n}$. It is well known that the eigenvalues of a matrix are continuous functions of the entries of the matrix (see Kato, 2012, Ch. 2). Hence, it follows that pointwise convergence of a sequence of kernels k_{θ_i} to k implies pointwise convergence of the vectors of explained variances $\left(\mathcal{V}(f_{k_{\theta_i}}^1), \ldots, \mathcal{V}\left(f_{k_{\theta_i}}^n\right)\right)$ to $\left(\mathcal{V}\left(f_k^1\right), \ldots, \mathcal{V}\left(f_k^n\right)\right)$.

6.7 Mathematical Definition of General-Purpose Kernels

In summary, for each of the popular kernel methods reviewed, if a family of kernels $\mathcal{K}_\Theta$ is such that for any oracle kernel k we may always find a sequence $\{k_{\theta_i} \in \mathcal{K}_\Theta\}$ that converges pointwise to k, then we may always find an element $k_{\theta_N} \in \mathcal{K}_\Theta$ that performs as well as k within an arbitrarily small precision. This result may be generalised to any other kernel method that aims at minimising a loss function that depends on k through a continuous function of a finite number of evaluations $k(x_i, x_j)$. We may therefore define general-purpose kernels as follows.

Definition 6.1 *Let $\mathcal{X}$ and Θ be two metric spaces. We say of a family of real-valued positive semi-definite functions $\mathcal{K}_\Theta = \{k_\theta : \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \theta \in \Theta\}$ that it is general-purpose when $\mathcal{K}_\Theta$ is dense in $\mathcal{K}_{CB}$, the family of all real-valued continuous bounded positive semi-definite functions on $\mathcal{X} \times \mathcal{X}$, with respect to the pointwise convergence of*

functions. That is, $\forall k \in \mathcal{K}_{CB}$*,*

$$\exists k_{\theta_i} \in \mathcal{K}_\Theta, \ \ s.t. \ \forall u, v \in \mathcal{X}, k_{\theta_i}(u, v) \underset{i \to +\infty}{\longrightarrow} k(u, v). \tag{6.5}$$

Without loss of generality, in the following Chapter we take $\mathcal{X} = \mathbb{R}^d$, and we denote $\mathcal{B}(\mathbb{R}^d)$ the Borel σ-algebra on $\mathbb{R}^d$.

Chapter 7

Some Tractable General-Purpose Families of Kernels

> "There is no such a thing as data. What is commonly called data are a combination of a single datum and some representation."
>
> *Terry Lyons*

Having formalised what we denote general-purpose families of kernels, we now move on to constructing such families. Given the wide adoption of stationary kernels in the machine learning community, we begin by proposing tractable families of stationary kernels that are pointwise dense in the family of all continuous stationary kernels. We will then extend these and propose tractable families of kernels that are general-purpose (i.e. pointwise dense not only in the family of continuous stationary kernels, but also in the space of continuous bounded nonstationary kernels).

Some of the results on flexible stationary kernels we present in this Chapter were previously discussed in Chapter 3 (Section 3.3.1). We choose to recall them here in the interest of keeping Part III self-contained.

7.1 Mathematical Preliminaries

A crucial result on the characterisation of stationary kernels is Bochner's theorem (Rasmussen and Williams (2005); Rudin (1962); Stein (1999)).

Theorem 7.1 *(Bochner's theorem) A complex-valued continuous function k on $\mathbb{R}^d$ is a stationary kernel if and only if it can be represented as*

$$k(\tau) = \int_{\mathbb{R}^d} e^{2\pi i \omega^T \tau} \mu(d\omega), \tag{7.1}$$

where μ is a positive finite measure.

Bochner's theorem introduces a duality between the class of stationary kernels and the class of positive finite measures μ in the spectral domain. This result is well complemented by the notion of weak convergence of measures (Bergström (2014)) which we recall below.

Definition 7.2 *A sequence of measures μ_n on $\left(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d)\right)$ weakly converges to a measure μ on $\left(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d)\right)$ when for all continuous bounded functions f,*

$$\int_{\mathbb{R}^d} f(\omega)\mu_n(d\omega) \underset{n\to+\infty}{\longrightarrow} \int_{\mathbb{R}^d} f(\omega)\mu(d\omega). \tag{7.2}$$

In particular, it immediately follows from the boundedness of the complex exponential in Equation (7.1) that, if a sequence of positive finite measures μ_n weakly converges to another positive finite measure μ, then for every $\tau \in \mathbb{R}^d$ the sequence $k_n(\tau) := \int_{\mathbb{R}^d} e^{2\pi i \omega^T \tau} \mu_n(d\omega)$ converges to $k(\tau) := \int_{\mathbb{R}^d} e^{2\pi i \omega^T \tau} \mu(d\omega)$. In other words, a sufficient condition for a sequence of complex-valued stationary kernels k_n to be pointwise dense in the family of all complex-valued continuous stationary kernels is that the corresponding family of spectral measures μ_n be weakly dense in the family of all positive finite measures. Moreover, the corresponding sequence of real-parts $\mathcal{R}e(k_n)$, which are also positive semi-finite functions,[1] are dense in the family of all real-valued continuous stationary kernels.

Finally, it is well-known that positive finite discrete (or pure-point) measures, that is, positive finite measures supported on a countable set, are weakly dense in the space of all positive finite measures (Hu and Papageorgiou, 2013, Lemma 2.9). Recalling that all positive finite discrete measures on $\left(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d)\right)$ are of the form

$$\mu_{\text{sing.}} = \sum_{k=1}^{+\infty} a_k \delta_{\{\omega_k\}},$$

[1] For a proof see Loève (1963) p. 133.

where $\omega_k \in \mathbb{R}^d, \quad a_k \geq 0, \quad \sum_{k=1}^{+\infty} a_k < +\infty$, and

$$\forall A \subset \mathbb{R}^d, \quad \delta_{\{x\}}(A) = \begin{cases} 1 & \text{if } x \in A \\ 0 & \text{if } x \notin A \end{cases},$$

it follows from applying the results discussed above that the family of complex-valued stationary kernels of the form

$$k_{\text{CSS}}(\tau; K) = \sum_{k=1}^{K} a_k e^{2\pi i \omega_k^T \tau}, \quad a_k \geq 0, \quad \omega_k \in \mathbb{R}^d, \tag{7.3}$$

whose spectral measures are $\mu_{\text{CSS}} := \sum_{k=1}^{K} a_k \delta_{\omega_k}$, are pointwise dense in the family of all complex-valued continuous stationary kernels on $\mathbb{R}^d$. Consequently, the family of real-parts

$$\mathcal{R}e\left(k_{\text{CSS}}(\tau; K)\right) = \sum_{k=1}^{K} a_k \cos\left(2\pi \omega_k^T \tau\right), \tag{7.4}$$

are pointwise dense in the family of all real-valued continuous stationary kernels on $\mathbb{R}^d$. These kernels are a slight generalisation of the *sparse spectrum* kernels (Lazaro-Gredilla et al. (2010)), where the authors restricted themselves to equal coefficients of the form $a_k = \frac{\sigma^2}{K}$. Kernels of the form of Equation (7.3) also arise in *random Fourier features* methods (Le et al. (2013); Rahimi and Recht (2007); Yang et al. (2015)) that are concerned with approximating a known kernel with one that has a finite-dimensional feature space in order to speed-up inference. They are based on the observation that Equation (7.1) may be rewritten as

$$k(\tau) = \sigma^2 \int_{\mathbb{R}^d} e^{2\pi i \omega^T \tau} \mathbb{P}(d\omega) := \sigma^2 \mathrm{E}_{\mathbb{P}}\left(e^{2\pi i \omega^T \tau}\right) \approx \frac{\sigma^2}{n} \sum_{j=1}^{n} e^{2\pi i \omega_j^T \tau} \tag{7.5}$$

with $\mathbb{P} := \frac{\mu}{\mu\left(\mathbb{R}^d\right)}$, $\sigma^2 := \mu\left(\mathbb{R}^d\right)$ and where ω_j are i.i.d. drawn from the probability distribution $\mathbb{P}$, which results in a consistent and unbiased estimate of $k(\tau)$.

7.2 Stationary Generalized Spectral Kernels

Despite the density property, kernels of the form of Equation (7.4) have some practical pitfalls. Firstly, they are periodic and infinitely differentiable, and consequently the RKHSs they induce are families of such functions. In fact, the corresponding RKHSs

can be shown to consist of linear combinations of trigonometric functions with up to K different frequencies. This excess regularity might require a large number of spectral components K to flexibly cope with non-periodic patterns in datasets, and a large K would result in more costly and less robust inference when the parameters a_k and ω_k are inferred directly. Secondly, these kernels are not integrable. When used in Gaussian process regression for instance, the covariance between the values of the Gaussian process at two points will not vanish as the distance between the points becomes arbitrarily large, which would impose *a priori* the view that the underlying function is highly structured, which might be unrealistic in many real-life non-periodic applications, unless the input space is bounded and the period is much larger than the diameter of the input space. Finally, it is preferable for practical purposes that families of stationary kernels be proposed that are pointwise dense in the family of continuous stationary kernels, but also have popular kernels as elementary special cases rather than limit cases.[2]

Definition 7.3 *We denote stationary generalized spectral kernel any function of the form*

$$k_{SGS}(\tau; K) := \sum_{k=1}^{K} \alpha_k h(\tau \odot \gamma_k) \cos(2\pi \omega_k^T \tau), \tag{7.6}$$

with $\omega_k \in \mathbb{R}^d$, $\gamma_k \in \mathbb{R}^{+d}$, $\alpha_k \geq 0$, $K \in \mathbb{N}^*$, *and where* h *is a continuous and integrable stationary kernel.*

The parameters γ_k serve as inverse input scales. Moreover, kernels of the form of Equation (7.4) are recovered with $\gamma_k = 0$. We also note that $k_{\text{SGS}}(\tau; K)$ is a continuous stationary kernel as sum of products of such functions. Furthermore, the pointwise density is preserved as stated in the following theorem.

Theorem 7.4 *For every continuous and integrable stationary kernel* h*, the family of kernels*

$$\left\{ k_{SGS}(\tau; K), \quad \omega_k \in \mathbb{R}^d, \quad \alpha_k \geq 0, \quad \gamma_k \in \mathbb{R}^{+d}, \quad K \in \mathbb{N}^* \right\},$$

as defined in Definition 7.3, is dense in the family of all real-valued continuous stationary kernels with respect to the pointwise convergence of functions.

[2]More generally, it is worth noting that not all families of kernels that are pointwise dense in the space of continuous stationary kernels are equally efficient. As illustred in the foregoing paragraph, some families might require a larger model complexity than others to achieve the same performance.

Proof The subfamily $\mathcal{R}e\left(k_{\text{CSS}}(\tau; K)\right)$, which corresponds to $\gamma_k = 0$ and $a_k = \alpha_k h(0)$, was previously shown to be pointwise dense in the family of all continuous stationary kernels, and consequently so is $\{k_{\text{SGS}}(\tau; K)\}$. ■

It is worth noting that the density property of Theorem 7.4 is preserved even when γ_k is required to have strictly positive coordinates, which for instance will be the case if one chooses to parameterize the kernel through input scales rather than inverse input scales. A few properties of stationary generalized spectral kernels are worth stressing at this point.

Remarks: A mean zero stationary Gaussian process with *stationary generalized spectral kernel* as covariance function is p times continuously differentiable in the mean square sense if and only if a mean zero stationary Gaussian process with covariance function h is (see Appendix D.2 for a proof). Moreover, a stationary generalized spectral kernel is integrable if and only if $\forall k \leq K$, either $\gamma_k \neq 0$ or $\alpha_k = 0$ (see Appendix D.3 for a proof).

Examples: We have already seen that the *sparse spectrum* kernels correspond to the case $\gamma_k = 0$ with equal α_k terms. Moreover, the *spectral mixture kernel* of Wilson and Adams (2013) corresponds to the special case $h(\tau) = \exp(-2\pi^2||\tau||^2)$, and yields infinitely differentiable GPs as a result of the above proposition. It is easy to verify that the Matérn kernels

$$k_{\text{MA}}(\tau; \nu) = \frac{1}{\Gamma(\nu)2^{\nu-1}} \left(||\tau||\sqrt{2\nu}\right)^{\nu} K_{\nu} \left(||\tau||\sqrt{2\nu}\right),$$

where Γ is the gamma function and K_ν is the modified Bessel function of second kind, satisfy the conditions set for h in Definition 7.3. Hence, *spectral Matérn kernels*

$$k_{\text{SGS-MA}}(\tau; \nu) = \sum_{k=1}^{K} \alpha_k k_{\text{MA}} \left(\tau \odot \gamma_k; \nu\right) \cos\left(2\pi\omega_k^T \tau\right), \tag{7.7}$$

are also pointwise dense in the family of continuous stationary kernels, and allow learning the differentiability of the underlying latent function from the data. It results from the foregoing remarks that this may be done by considering values of ν of the form $\frac{1}{2} + i, i \in \mathbb{N}$, which corresponds to continuity for $i = 0$ and i times continuous differentiability for $i > 0$; this is standard practice for the vanilla Matérn kernel. As noted by Rasmussen and Williams (2005), values $i > 2$ usually perform identically to

$i = 2$ so that for all practical purposes, learning differentiability may be thought of as comparing the kernels $k_{\text{SGS-MA}}\left(\tau; \frac{1}{2}\right)$, $k_{\text{SGS-MA}}\left(\tau; \frac{3}{2}\right)$, and $k_{\text{SGS-MA}}\left(\tau; \frac{5}{2}\right)$.

7.3 Spectral Characterisation of Continuous Bounded Kernels

Bochner's theorem was the cornerstone of the previous section. The spectral characterisation of stationary kernels it provides turned the problem of pointwise convergence of stationary kernels into that of weak convergence of measures in the spectral domain. Luckily, a larger class of kernels exists that admits a spectral characterisation similar to Bochner's theorem, namely *harmonizable covariance functions*, which were first introduced by Loève (1963) and have subsequently been generalized and extensively studied in Kakihara (1985) and Kakihara (1997). We recall some important notions.

Definition 7.5 *(Weak harmonizability) A complex-valued function k on $\mathbb{R}^d \times \mathbb{R}^d$ is said to be a weakly harmonizable covariance function when it can be represented as*

$$k(x, y) = \int_{\mathbb{R}^d \times \mathbb{R}^d} e^{2\pi i\left(\omega_1^T x - \omega_2^T y\right)} \mu_F(d\omega_1, d\omega_2), \tag{7.8}$$

where μ_F is a (real-valued) positive definite bimeasure on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d)) \times (\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$ of bounded semivariation (or Fréchet variation), and the integral is in the Morse-Transue sense (Morse (1955)). In other words, μ_F satisfies the following conditions:

- *(Bimeasure) $\forall A, B \in \mathcal{B}(\mathbb{R}^d)$, the maps $B \to \mu_F(A, B)$ and $B \to \mu_F(B, A)$ define measures on $(\mathbb{R}^d, \mathcal{B}(\mathbb{R}^d))$.*
- *(Positive definiteness)*

$$\forall \alpha_1, \ldots, \alpha_n \in \mathbb{C},\ A_1, \ldots, A_n \in \mathcal{B}(\mathbb{R}^d), \quad \sum_{i=1}^{n} \sum_{j=1}^{n} \overline{\alpha_i} \mu_F(A_i, A_j)\ \alpha_j \geq 0. \tag{7.9}$$

- *(Fréchet variation boundedness) $||\mu_F||\left(\mathbb{R}^d, \mathbb{R}^d\right) < \infty$, where*

$$\forall A, B \in \mathcal{B}(\mathbb{R}^d),\ ||\mu_F||(A, B) := \sup \left| \sum_{i=1}^{n} \sum_{j=1}^{m} \overline{\alpha_i} \mu_F(A_i, B_j) \beta_j \right|, \tag{7.10}$$

the sup *being taken over all finite measurable partitions of A and B, namely $(A_i)_{1 \leq i \leq n}$ and $(B_j)_{1 \leq j \leq m}$, and over all $\alpha_i, \beta_j \in \mathbb{C}$ such that $|\alpha_i|, |\beta_j| \leq 1$.*

Definition 7.6 *(Strong harmonizability) A strongly harmonizable covariance function is a weakly harmonizable covariance function whose spectral bimeasure μ_F is of finite total (Vitali) variation, that is $|\mu_F|\left(\mathbb{R}^d, \mathbb{R}^d\right) < \infty$, where*

$$\forall A, B \in \mathcal{B}(\mathbb{R}^d),\ |\mu_F|(A, B) := \sup \sum_{i=1}^{n} \sum_{j=1}^{m} |\mu_F(A_i, B_j)|\,, \tag{7.11}$$

the sup *being taken over all finite measurable partitions of A and B, namely $(A_i)_{1\leq i\leq n}$ and $(B_j)_{1\leq j\leq m}$.*

Remarks: By applying the definition of positive definiteness of a real-valued bimeasure μ_F (Equation (7.9)) with $\alpha_0 = 1,\ \alpha_1 = i$, it follows that

$$\forall A \in \mathcal{B}(\mathbb{R}^d),\quad \mu_F(A, A) \geq 0$$

and

$$\mu_F(A_0, A_0) + \mu_F(A_1, A_1) + i(\mu_F(A_0, A_1) - \mu_F(A_1, A_0)) \in \mathbb{R},$$

which implies in particular that every positive definite (real-valued) bimeasure is symmetric:

$$\forall A_0, A_1 \in \mathcal{B}(\mathbb{R}^d),\quad \mu_F(A_0, A_1) = \mu_F(A_1, A_0).$$

When the spectral bimeasure μ_F is of finite total variation (i.e. in the event of strong harmonizability), μ_F turns out to be a fully-fledged measure on $(\mathbb{R}^d \times \mathbb{R}^d, \mathcal{B}(\mathbb{R} \times \mathbb{R}^d))$ and the Morse-Transue integral coincides with the Lebesgue integral on $\mathbb{R}^d \times \mathbb{R}^d$ (see Mehlman (2004) p. 50 or Kakihara (1997) p. 4).

It is worth noting that every function k that can be represented as in Equation (7.8), where the positive definite bimeasure μ_F is either of finite total variation or of finite total semivariation, is indeed a continuous bounded kernel.[3] Moreover, it follows from Bochner's theorem that every continuous stationary kernel is strongly harmonizable and the corresponding spectral bimeasure has mass concentrated along the diagonal $\omega_1 = \omega_2$. However, harmonizable covariance functions constitute a much bigger class than stationary covariance functions. In fact, it is commonly said that all continuous bounded kernels of practical interest are indeed strongly harmonizable. For instance, Yaglom noted that the only continuous bounded kernels known to the author that are not (strongly) harmonizable 'are rather complicated and have some unusual,

[3]The continuity of k is easily proved from first principles by applying the dominated convergence theorem to Equation (7.8). The symmetry and positive semi-definitiveness of k are direct consequences of the symmetry and positive definiteness of μ_F. Finally, the boundedness of k is obtained by noting that $|k(x, y)| \leq ||\mu_F||(\mathbb{R}^d, \mathbb{R}^d) < \infty$.

even pathological properties' (Yaglom, 1987, p. 464). More recently, Genton said of the characterisation of Equation (7.5) that it holds for all continuous bounded kernels (Genton, 2002, p. 308). This intuition that we might not be missing out by restricting ourselves to strongly harmonizable kernels has been formalised by Dehay and Moché (1986), who studied the difference between the classes of weakly harmonizable kernels, strongly harmonizable kernels, and continuous bounded kernels. We recall their main result below.

Theorem 7.7 *Every complex-valued continuous bounded kernel defined on $\mathbb{R}^d \times \mathbb{R}^d$ is the limit, uniformly on compact subsets of $\mathbb{R}^d \times \mathbb{R}^d$, of a sequence of strongly harmonizable covariance functions, each of them admitting an absolutely continuous spectral measure.*

Recalling that uniform convergence implies pointwise convergence, a direct consequence of the theorem above is stated as follows.

Corollary 7.8 *The family of strongly harmonizable kernels defined on $\mathbb{R}^d \times \mathbb{R}^d$ is pointwise dense in the family of all complex-valued continuous bounded kernels defined $\mathbb{R}^d \times \mathbb{R}^d$.*

7.4 General-Purpose Spectral Kernels

It follows from Corollary 7.8 that, in order for a family of kernels to be general-purpose, or equivalently pointwise dense in the family of all continuous bounded kernels, it suffices that it be pointwise dense in the family of all strongly harmonizable covariance functions, so that we may focus on the latter property.

7.4.1 Intuition

The spectral characterisation Equation (7.8) generalises our previous observation that weak convergence of spectral measures implies pointwise convergence of kernels to strongly harmonizable kernels. Moreover, we note that the spectral measure μ_F of a strongly harmonizable kernel k is finite as

$$\int_{\mathbb{R}^d \times \mathbb{R}^d} \mu_F(d\omega_1, d\omega_2) = k(0,0) < \infty,$$

and we prove in Appendix D.4 that finite *discrete* positive definite bimeasures are weakly dense in the family of all finite positive definite bimeasures. Interestingly,

the weak density is preserved when the above discrete bimeasures are restricted to the ones that arise as a finite sum of bimeasures with disjoint supports of the form $\{\omega_k^1, \omega_k^2\} \times \{\omega_k^1, \omega_k^2\}$, which we prove in Appendix D.4. These spectral measures yield (pointwise dense) trigonometric kernels of the form

$$k_{\text{CNS}}(x, y; K) = \sum_{k=1}^{K} \Upsilon_k(x)^* \Upsilon_k(y), \tag{7.12}$$

where $\Upsilon_k(x) := U_k \begin{pmatrix} e^{2\pi i x^T \omega_k^1} \\ e^{2\pi i x^T \omega_k^2} \end{pmatrix}$, with U_k any 2×2 upper triangular matrix, and where $\Upsilon_k^*(x)$ is the conjuguate transpose of $\Upsilon_k(x)$. We note that the stationary equivalent, k_{CSS} of Equation (7.3), corresponds to the special case where $U_k = \begin{pmatrix} \frac{\alpha_k}{2} & 0 \\ 0 & \frac{\alpha_k}{2} \end{pmatrix}$ and $\omega_k^1 = \omega_k^2$. Consequently, the sequence of real-parts

$$\begin{aligned} k_{\text{NSS}}(x, y; K) :=& \mathcal{R}e\left(k_{\text{CNS}}(x, y; K)\right) \\ =& \sum_{k=1}^{K} \Psi_k(x)^T \Psi_k(y), \end{aligned} \tag{7.13}$$

with

$$\Psi_k(x) = \begin{pmatrix} U_k \begin{bmatrix} \cos(2\pi x^T \omega_k^1) \\ \cos(2\pi x^T \omega_k^2) \end{bmatrix} \\ \\ U_k \begin{bmatrix} \sin(2\pi x^T \omega_k^1) \\ \sin(2\pi x^T \omega_k^2) \end{bmatrix} \end{pmatrix},$$

is pointwise dense in the family of all strongly harmonizable kernels, and therefore also in the family of all continuous bounded kernels. We then extend these kernels in order to allow for integrability as we did in the stationary case to obtain *generalized spectral kernels*, which we define as follows.

7.4.2 The General Case

Definition 7.9 *Let $\{g_\rho, \rho \in \mathbb{R}^{+p}\}$ with $p \in \mathbb{N}^*$ be any parametric family of continuous kernels such that g_{ρ^*} is a constant function for some $\rho^* \in \mathbb{R}^{+p}$. We denote generalized spectral kernel any function of the form:*

$$k_{GS}(x, y; K) = \sum_{k=1}^{K} g_{\rho_k}(x, y) \Psi_k(x)^T \Psi_k(y), \tag{7.14}$$

where $K \in \mathbb{N}^*$, *and*

$$\Psi_k(x) = \begin{pmatrix} U_k \begin{bmatrix} \cos(2\pi x^T \omega_k^1) \\ \cos(2\pi x^T \omega_k^2) \end{bmatrix} \\ \\ U_k \begin{bmatrix} \sin(2\pi x^T \omega_k^1) \\ \sin(2\pi x^T \omega_k^2) \end{bmatrix} \end{pmatrix}$$

with $U_k \in \mathcal{U}$, $\mathcal{U}$ *the set of all* 2×2 *upper triangular matrices, and* $\omega_k^1,\ \omega_k^2 \in \mathbb{R}^d$.

When $U_k = \begin{pmatrix} \frac{\alpha_k}{2} & 0 \\ 0 & \frac{\alpha_k}{2} \end{pmatrix}$ and $\omega_k^1 = \omega_k^2$ we have that

$$\Psi_k(x)^T \Psi_k(y) = \alpha_k \cos\left(2\pi(x-y)^T \omega_k^1\right),$$

and we recover *stationary generalized spectral* kernels as a special case of *generalized spectral kernels* when additionally $p = d$ and $g_\rho(x,y) = h\left((x-y) \odot \rho\right)$ with h as in Definition 7.3. Indeed, we may choose $\rho^* = 0$ given that $\forall x, y,\ g_0(x,y) = h(0)$. Moreover, we note that *generalized spectral kernels* are indeed continuous kernels as sum of products of such functions.[4] Furthermore, they are general-purpose as stated in the following theorem.

Theorem 7.10 *For every parametric family of continuous kernels* $\{g_\rho, \rho \in \mathbb{R}^{+p}\}$ *with* $p \in \mathbb{N}^*$ *such that* g_{ρ^*} *is a constant function for some* $\rho^* \in \mathbb{R}^{+p}$, *the corresponding family of generalized spectral kernels*

$$\left\{k_{GS}(x,y;K),\ \omega_k^1, \omega_k^2 \in \mathbb{R}^d,\quad U_k \in \mathcal{U},\quad \rho_k \in \mathbb{R}^{+p},\quad K \in \mathbb{N}^*\right\},$$

as defined in Definition 7.9, is general-purpose (i.e. dense in the family of all real-valued continuous bounded kernels with respect to the pointwise convergence of functions).

Proof See Appendix D.4. ■

Remarks: The differentiability and integrability discussions of the stationary case are easily extended. In the general case, the degree of smoothness of a mean zero GP with covariance function k_{GS} is determined by the kernels g_{ρ_k} and may once again be learned for instance using the Matérn family of kernels. As for the integrability of $k_{\text{GS}}(.,.;K)$ (as a function defined on $\mathbb{R}^d \times \mathbb{R}^d$), it is equivalent to the integrability of all

[4] The function $(x,y) \to \Psi_k(x)^T \Psi_k(y)$ is easily proven to be positive semi-definite from first principles.

the functions $\{g_{\rho_k}, k \leq K\}$ regarded as functions defined on $\mathbb{R}^d \times \mathbb{R}^d$, which requires in particular that $\rho^* \notin \{\rho_k, k \leq K\}$ and that no g_{ρ_k} be a stationary kernel.

7.4.3 Hybrid Generalized Spectral Kernels

A form of *generalized spectral kernels* that is of particular practical interest arises when the family g_ρ contains both stationary and nonstationary kernels, so that 'nonstationarity' (i.e whether or not the latent function learned by our machine learning task should be invariant by translation of the training inputs) may effectively be learned from the data. An example of such *generalized spectral kernels* are *hybrid generalized spectral kernels* defined as follows.

Definition 7.11 *We denote hybrid generalized spectral kernels any generalized spectral kernel as defined in Definition 7.9 with g_ρ that is of the form*

$$g_\rho(x, y) = h\left((x - y) \odot \gamma\right) \eta(x \odot \kappa, y \odot \kappa), \tag{7.15}$$

where $\rho = (\gamma, \kappa) \in \mathbb{R}^{+2d}$, h is any continuous stationary integrable (as a function defined on $\mathbb{R}^d$) kernel, and η is any continuous nonstationary integrable (as a function defined on $\mathbb{R}^d \times \mathbb{R}^d$) kernel.

It follows from Theorem 7.10 that *hybrid generalized spectral kernels* (HGSKs) are also general-purpose. Noting that stationary kernels correspond to the special case $\kappa_k = 0,\ \omega_k^1 = \omega_k^2$, it follows that HGSKs allow learning nonstationarity from the data through the parameter κ. As h is bounded by $h(0)$, a necessary and sufficient condition for a HGSK to be integrable is that $\forall k \leq K,\ U_k = 0$ or $\kappa_k \neq 0$. Moreover, the degree of differentiability may once again be controlled or learned through h as previously discussed.

Examples of HGSKs

Silverman (1957) introduced numerous examples of real-valued continuous and integrable nonstationary kernels that are so-called 'locally stationary'; i.e. of the form

$$\eta(x, y) = k_1\left(x - y\right) k_2\left(\frac{x + y}{2}\right), \tag{7.16}$$

where k_1 is a stationary kernel and k_2 is positive-valued. Such a choice of η would be appropriate for learning local stationarity as an alternative to stationarity, with

the theoretical guarantee of asymptotically performing as well as any other form of nonstationarity as the number of spectral components K increases.

η may also be chosen to be of the form

$$\eta(x, y) = k_1(x)k_1(y), \tag{7.17}$$

where k_1 is any continuous, integrable real-valued function. This choice of η is particularly interesting as it enables scalable inference through finite-dimensional feature space approximations. In effect the resulting kernel can be written as

$$k_{\text{HGS}}(x, y) = \sum_{k=1}^{K} h\left((x - y) \odot \gamma_k\right) \Phi_k(x)^T \Phi_k(y), \tag{7.18}$$

where $\Phi_k(x) = k_1(x \odot \kappa_k)\Psi_k(x)$. Thus, any consistent random Fourier approximation of h (Rahimi and Recht (2007)), $h(x - y) \approx \hat{H}(x)^T \hat{H}(y)$, would result in the consistent finite-dimensional feature space approximation

$$k_{\text{HGS}}(x, y) \approx \hat{k}_{\text{HGS}} := \sum_{k=1}^{K} \hat{G}_k(x)^T \hat{G}_k(y), \tag{7.19}$$

with $\hat{G}_k(x) = \hat{H}(x \odot \gamma_k) \otimes \Phi_k(x)$, where $\otimes$ denotes the Kronecker product. The speed of convergence of the resulting approximation is easily derived from that of the approximation of the stationary kernel h, several results on which are available in the literature (Rahimi and Recht (2007)). This approximation of HGSKs allows for inference in linear time complexity through the Woodbury matrix identity and the matrix determinant lemma. This is very important considering that HGSKs are general-purpose.

7.5 Numerical Evidence of General-Purposeness

We illustrate the pointwise density of general-purpose kernels by considering approximating popular nonstationary kernels with various spectral kernels, each with $K = 5$ spectral components. The nonstationary kernels we aim at approximating are: the linear kernel

$$k_{\text{Lin.}}(x, y) = (1 + xy),$$

the quadratic kernel

$$k_{\text{Quad.}}(x, y) = (1 + xy)^2,$$

the cubic kernel

$$k_{\text{Cub.}}(x, y) = (1 + xy)^3,$$

the hyperbolic tangent kernel

$$k_{\text{Tanh}}(x, y) = \tanh(1 + xy),$$

the neural network kernel

$$k_{\text{NN}}(x, y) = \frac{2}{\pi} \arcsin\left(\frac{2xy}{1 + 2xy}\right),$$

the Gibbs kernel

$$k_{\text{Gibbs}}(x, y) = e^{-\left(\frac{(x-y)^2}{e^{2x}+e^{2y}}\right)} \sqrt{\frac{2e^x e^y}{e^{2x} + e^{2y}}},$$

and fractional Brownian motion kernels

$$k_{\text{FBM}}(x, y; h) = \frac{1}{2}\left(|x|^{2h} + |y|^{2h} - |x - y|^{2h}\right),$$

with Hurst indices $h = i/10,\ i \in \{2, 4, 5, 6, 8\}$, the Brownian motion kernel being the special case $i = 5$. The parameters of the approximating spectral kernels are fitted by minimising the sum of the squared errors on the uniform grid $[0, 1] \times [0, 1]$ with mesh size 0.02. The resulting root mean square errors (RMSE) expressed as fraction of the average value of the true kernel are reported in Table 7.1. In Figure 7.1 we illustrate a section of fitted spectral kernels for the Brownian motion kernel, and the fractional Brownian motion kernels with Hurst indices 4/10 and 6/10.

We note from Table 7.1 that nonstationary spectral kernels considerably outperform stationary alternatives such as the *spectral mixture kernel* and the *sparse spectrum kernel*. More importantly, with only $K = 5$ spectral components, every nonstationary kernel considered was recovered almost perfectly by all *generalized spectral kernels*, which empirically validates the 'general-purposeness' we have proved. Furthermore, Figure 7.1 illustrates a limitation of *stationary general spectral kernels* in the experiment, namely that no matter K, they do not allow for differences between $k(0.5, 0.5 - u)$ and $k(0.5, 0.5 + u)$.

Table 7.1 Normalised RMSE of approximations of some nonstationary kernels by spectral kernels, each with $K = 5$ spectral components. S-SE is the spectral mixture kernel, SS is the sparse spectrum kernel, NSS is the nonstationary generalisation of sparse spectrum kernels (as per Equation (7.13)), and NS-[x] is the HGSK with η as in Equation (7.17) and where h and k_1 are of type [x].

	Lin.	Quad.	Cub.	Tanh	NN	Gibbs	FBM-2	FBM-4	BM	FBM-6	FBM-8
SS	0.26	0.52	0.78	0.04	0.62	0.04	0.35	0.53	0.60	0.65	0.74
S-SE	0.16	0.34	0.54	0.04	1.19	0.02	0.34	0.53	1.21	1.24	1.28
S-MA52	0.16	0.34	0.54	0.04	1.19	0.03	0.35	1.18	1.21	1.24	1.28
S-MA32	0.16	0.34	0.54	0.04	1.19	0.02	0.35	1.18	1.21	1.24	1.28
S-MA12	0.16	0.34	0.54	0.04	1.19	0.02	1.09	1.18	1.21	1.24	1.28
NSS	**0.00**	0.01	0.05	**0.01**	**0.00**	0.02	0.07	0.04	0.03	0.02	0.01
NS-SE	**0.00**	0.01	0.06	0.03	0.05	**0.00**	**0.05**	**0.02**	**0.01**	**0.01**	**0.00**
NS-MA52	**0.00**	0.01	0.06	0.03	0.05	0.02	0.07	0.04	0.03	0.02	0.01
NS-MA32	**0.00**	0.01	0.06	0.03	0.05	0.02	0.07	0.04	0.03	0.02	0.01
NS-MA12	**0.00**	**0.00**	**0.00**	0.02	0.04	**0.00**	**0.05**	**0.02**	**0.01**	**0.01**	**0.00**

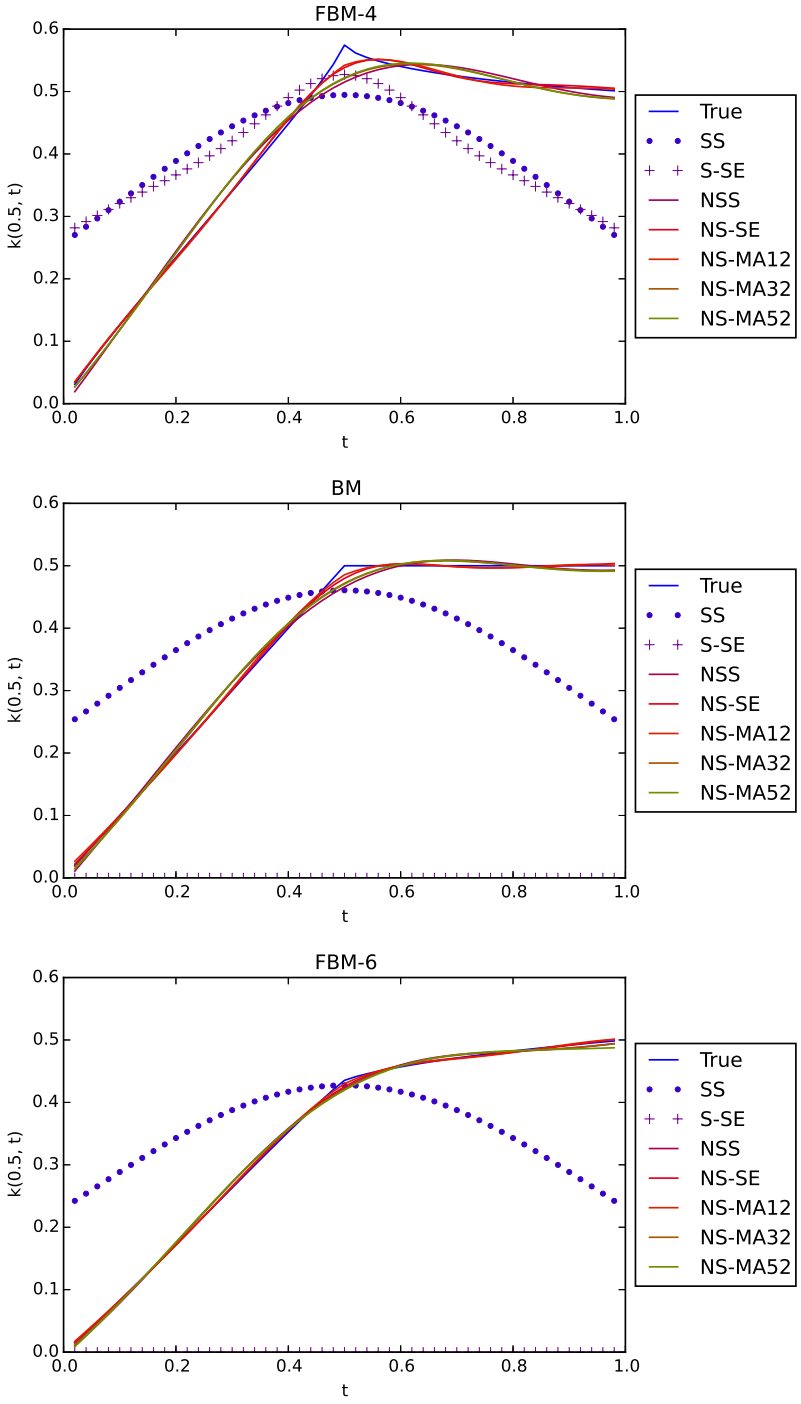

Fig. 7.1 Approximations of the Brownian motion kernel and fractional Brownian motion kernels with Hurst indices 4/10 and 6/10 on $[0,1] \times [0,1]$ by spectral kernels with $K = 5$ components.

7.6 Related Work

Approaches have been proposed in recent years that introduce greater flexibility by combining standard univariate kernels through series of compounded operations preserving the positive semi-definite property. Examples of such approaches include the *hierarchical kernel learning* model of Bach (2008), the *additive kernels* of Duvenaud et al. (2011) and the compositional search method of Duvenaud et al. (2013). Although these methods may be major improvements on popular isotropic kernels for some applications, the corresponding families of kernels are not dense in the family of stationary kernels. As for existing spectral kernels such as the sparse spectrum kernel (Lazaro-Gredilla et al. (2010)), the spectral mixture kernel (Wilson and Adams (2013)), and random Fourier kernels (Le et al. (2013); Rahimi and Recht (2007); Yang et al. (2015)) they are all special cases of *stationary generalized spectral kernels*. The *stationary generalized spectral kernels* we propose complement these approaches by allowing the learning of the degree of differentiability of the latent function while preserving the pointwise density in the space of all continuous stationary kernels.

Critically, unlike *generalized spectral kernels*, existing spectral kernels are all stationary. As for existing nonstationary kernels, commonly used approaches such as the input-dependent rescaling of stationary covariance functions of Paciorek and Schervish (2004) and spatial deformation of stationary covariance functions (Damian et al. (2001); Sampson and Guttorp (1992); Schmidt and O'Hagan (2003)), are application-specific, and they are not guaranteed to perform as well as every continuous bounded kernel. Considering that *generalized spectral kernels* arise as linear combinations of elementary kernels, they can be viewed as a special type of *multiple kernel learning* method (Gönen and Alpaydın (2011); Sonnenburg et al. (2006)). *Generalized spectral kernels* provide the additional theoretical guarantee that any continuous bounded kernel may indeed be 'learned' using the same family of base kernels, which is unprecedented in the *multiple kernel learning* literature: we believe this guarantee to be an important step in the development of fully automated kernel methods.

7.7 Summary

In Chapter 6 we argued that 'general-purpose' families of kernels should be families of kernels that are pointwise dense in the family of all continuous bounded kernels. This notion guarantees that, for a given dataset, a kernel exists in the general-purpose family that may perform as well as any oracle continuous bounded kernel. In this Chapter

we propose the first families of kernels that we prove are general-purpose, which we collectively refer to as *generalized spectral kernels* (GSKs). The density property is illustrated in an experiment where we show that commonly used nonstationary kernels are easily recovered by GSKs. We also propose example integrable general-purpose kernels that are amenable to consistent finite-dimensional feature space approximations. This opens the door to random Fourier approximation methods on a family of kernels that may be as flexible as required.

A big question remains open: how can we learn the required complexity of the kernel (i.e. the number K of spectral component) from the data? This will be the primary purpose of Chapter 8.

Chapter 8

Scalable Inference with General-Purpose Kernels

> "Everything must be made as simple as possible. But not simpler."
>
> *Albert Einstein*

In the previous two chapters we argued that for a family of kernels to be suitable for automated kernel learning, it is sufficient that it be pointwise dense in the family of all continuous bounded kernels (property we referred to as *general-purposeness*). This is due to the fact that, as we have proved in Chapter 6, pointwise convergence of kernels implies convergence of performances in kernel methods. Moreover, we have proposed tractable such families, namely *generalized spectral kernels*. In this chapter we propose inference techniques for scalable Bayesian nonparametric kernel learning from a *general-purpose* family. Our approaches have training time complexity and memory requirement that grow in $\mathcal{O}(nd)$ for n training samples and a d-dimensional input space, and test time complexity and memory requirement that grow in $\mathcal{O}(d)$. Crucially, our approaches enable the learning of *nonstationarity* and *model complexity* from the data.

8.1 Introduction

8.1.1 Related Work

Approaches have been proposed to scale-up kernel methods by exploiting the geometry of structured input spaces and kernels, for instance using Kronecker techniques when the input space forms a Cartesian grid and the kernel is separable (Saatchi (2011)), or using Toeplitz techniques when the input space is a one-dimensional uniform grid. Wilson and Nickisch (2015) proposed interpolating kernels when the input space does not have a suitable geometry so as to extend the scope of Kronecker and Toeplitz techniques. However, this extension still requires kernels to be separable and stationary. Random Fourier features methods (RFF) (Le et al. (2013); Rahimi and Recht (2007); Yang et al. (2015)) have also been proposed that scale up kernel methods by approximating a stationary kernel that has an infinite-dimensional feature space with one whose feature space has dimension, say m, negligible compared to the data size n, namely $m \ll n$. For instance, Rahimi and Recht (2007) proposed using the spectral characterisation of continuous stationary kernels provided by Bochner's theorem (Rudin (1962)) to construct the following unbiased and consistent approximation

$$k(x-y) = k(0) \int \cos\left(2\pi\omega^T(x-y)\right) \mathbb{P}_\omega(d\omega) \approx \frac{k(0)}{m} \sum_{i=1}^{m} \cos\left(2\pi\omega_i^T(x-y)\right) = \Phi(x)^T\Phi(y),$$

with $\Phi(x) = \sqrt{\frac{k(0)}{m}}\left(\ldots, \cos(2\pi\omega_i^T x), \ldots, \sin(2\pi\omega_i^T x), \ldots\right)$ and $\omega_i \overset{\text{i.i.d.}}{\sim} \mathbb{P}_\omega$. The computational bottleneck of this approximation is the evaluation of a vector of the form Ωx, where Ω is the matrix whose i-th row is the vector of random frequencies ω_i, which has time complexity $\mathcal{O}(md)$ where d is the dimension of the input space. In their so-called *Fastfood* approach, Le et al. (2013) proposed replacing the matrix Ω with a matrix V that arises as a product of structured and sparse matrices. The rows of the proposed matrix V turn out to be distributed as per $\mathbb{P}_\omega$, and consequently the resulting kernel approximation remains unbiased although, unlike Ω, V has correlated rows, which might result in a higher variance of the kernel approximation. The authors proposed exploiting the structure of V to compute the product Vx in $\mathcal{O}(m \log d)$. This work was further extended by the *À-la-carte* method of Yang et al. (2015) where, unlike the previous two works, the authors proposed learning $\mathbb{P}_\omega$ from a family of probability measures with densities that are isotropic and piecewise-linear in $||\omega||$. These RFF approaches often result in overall time complexity (resp. memory requirement) dropping from $\mathcal{O}(n^3)$ (resp. $\mathcal{O}(n^2)$) to $\mathcal{O}(nm^2)$ (resp. $\mathcal{O}(nm)$). However, Rahimi

and Recht (2007) and Le et al. (2013) assumed the kernel k to be given, although flexibly learning the shape of the kernel is arguably the most important step in kernel methods. As for Yang et al. (2015), the isotropy requirement imposed in the learning of k might be a major restriction in that it introduces a dependency to the units of input coordinates. To see why, if we set $d = 2$ for the sake of clarity, then it is easy to see that the two families of spectral densities $\left\{(\omega_1, \omega_2) \to \rho\left(\sqrt{\omega_1^2 + \omega_2^2}\right), \rho \in \text{PWL}\right\}$ and $\left\{(\omega_1, \omega_2) \to \rho\left(\sqrt{\omega_1^2 + 10\omega_2^2}\right), \rho \in \text{PWL}\right\}$, where PWL denotes the set of positive-valued piecewise-linear functions, are not the same. Hence, the hypothesis space of candidate kernels explored by Yang et al. (2015) depends on units in which inputs are expressed.

Approaches have been proposed that introduce greater flexibility by learning a kernel as a linear combination of simpler kernels belonging to a given dictionary of kernels (Gönen and Alpaydın (2011); Sonnenburg et al. (2006)), or by combining standard univariate kernels through series of compounded operations preserving the positive semi-definite property (Bach (2008); Duvenaud et al. (2013, 2011)). Although these methods may improve on vanilla alternatives such as the RBF kernel in certain applications, they provide no flexibility or performance guarantees.

The closest work to ours is the BaNK model of Oliva et al. (2016). The authors considered learning $\mathbb{P}_\omega$ as a Dirichlet process (DP) mixture of (fully-dependent and d-dimensional) Gaussians, which gives rise to the spectral mixture kernel of Wilson and Adams (2013).

Perhaps the most obvious criticism of their work is that, like most of the previously mentioned works, it is limited to stationary kernels.

The second criticism is that, although the spectral mixture kernel is pointwise dense in the space of continuous stationary kernels, when the number of random Fourier features m is fixed, the resulting family of trigonometric kernels are no longer pointwise dense in the family of continuous stationary kernels. Consequently, the flexibility of the Dirichlet process mixture prior over the spectral measure might be restricted when m is too small for the problem of interest, which is hard to determine beforehand. For instance, the Dirichlet process mixture cannot learn more than m coexisting clusters from m samples. One might be tempted to choose a very large m so as to improve the quality of the random Fourier approximation, and subsequently increase the flexibility of the spectral measure. However, this would come at the expense of scalability, as the overall time complexity is quadratic in m. Too large an m would unduly slow the inference down, and too small an m would reduce model-fit. Clearly, m should be learned so as to provide an appropriate trade-off between scalability or simplicity and

model-fit. Intuitively, an increase in used computational resources should only be the price to pay for a superior model-fit.

The reader might still be concerned that, given that the variance of the random Fourier approximation decreases with m, a small m would result in a poor Monte Carlo approximation, which the reader might think would not be suitable, regardless of whether m is learned or not. This argument would have been valid if the aim of BaNK was to accurately approximate a given spectral mixture kernel. However, this is not the case: BaNK aims at flexibly learning a stationary kernel from the data. A different perspective on BaNK, which might clarify any remaining doubts, is obtained by recalling that the family of kernels

$$k(x-y) = \frac{k(0)}{m} \sum_{i=1}^{m} \cos\left(2\pi\omega_i^T(x-y)\right), \tag{8.1}$$

which is parameterised by m, $k(0)$ and ω_i, are flexible enough to learn any stationary kernel,[1] and by noting that BaNK corresponds to placing as regularising prior over the frequencies ω_i a Dirichlet process mixture of multivariate Gaussians, and fixing m. We stress that, in this perspective, the kernel of Equation (8.1) is not introduced as an approximation of the spectral mixture kernel, although the average kernel under this prior specification (for a fixed m) is effectively a spectral mixture kernel. Thus, whether m should be small or large, has nothing to do with approximating the spectral mixture kernel, but is solely a question of what model complexity is warranted by the data. Hence, m should be learned.

The third criticism of the BaNK model is that it has time complexity (resp. memory requirement) cubic (resp. quadratic) in the input dimension, which is the result of allowing the multivariate Gaussians in the Dirichlet process mixture to be fully-dependent. This makes BaNK almost unusable on datasets with very large input dimensions such as images. The rationale provided by the authors for learning the spectral density as a mixture of fully-dependent Gaussians is that these are universal approximators of probability distributions. However, mixtures of multivariate Gaussians with *independent coordinates* are also universal approximators of probability distributions, so that allowing for full-dependence does not present obvious advantages, despite being computationally costlier.

[1] i.e. they are pointwise dense in the family of all continuous stationary kernels. To see why, we note that by denoting J the number of unique frequency vectors, and by denoting m_j the multiplicity of the frequency ω_j, we may rewrite Equation (8.1) as $k(x-y) = \sum_{j=1}^{J} \sigma_j^2 \cos\left(2\pi\omega_j^T(x-y)\right)$, where $\sigma_j^2 := k(0)\frac{m_j}{m}$. Finally, as the multiplicities m_j and the variance $k(0)$ vary, terms σ_j^2 may independently take any non-negative values, and the pointwise density follows from the results of Chapter 7.

The last but easiest to address criticism of BaNK is that the authors do not discuss the learning of the concentration parameter of the Dirichlet process, which may affect the learned number of clusters in the mixture. This can be addressed using standard techniques such as the one proposed by Escobar and West (1995) and Neal (2000).

Empirical evidence that nonstationary kernels may indeed outperform stationary ones in certain applications abound in the literature (see for instance Damian et al. (2001); Paciorek and Schervish (2004); Sampson and Guttorp (1992); Schmidt and O'Hagan (2003); Wilson et al. (2016)). However, proposed nonstationary kernels are often application-specific (Damian et al. (2001); Paciorek and Schervish (2004); Sampson and Guttorp (1992); Schmidt and O'Hagan (2003)) or, in the case of Wilson et al. (2016), with a number of parameters in the hundred of thousands irrespective of the dataset, the approach is prone to overfitting. The *general-purpose* kernels we have introduced in the previous chapter, namely *generalized spectral kernels* (GSKs), provide a flexible and principled alternative to the nonstationary kernels previously mentioned. The aim of this chapter is to introduce efficient methods for learning *generalized spectral kernels* in a wide variety of machine learning tasks.

8.1.2 GSKs for Nonstationarity Learning

We recall that *hybrid generalized spectral kernels* (HGSKs) are functions of the form

$$k_{\mathrm{HGS}}(x, y; K) = \sum_{k=1}^{K} h\left((x-y)\odot\gamma_k\right) k_1(x\odot\kappa_k)k_1(y\odot\kappa_k)\Psi_k(x)^T\Psi_k(y), \tag{8.2}$$

where $\omega_k^1, \omega_k^2 \in \mathbb{R}^d$, $\gamma_k, \kappa_k \in \mathbb{R}^{+d}$, $\Psi_k(x) = \begin{pmatrix} U_k \begin{bmatrix} \cos(2\pi x^T\omega_k^1) \\ \cos(2\pi x^T\omega_k^2) \end{bmatrix} \\ \\ U_k \begin{bmatrix} \sin(2\pi x^T\omega_k^1) \\ \sin(2\pi x^T\omega_k^2) \end{bmatrix} \end{pmatrix}$, with U_k any 2×2 upper triangular matrix, and where h is any continuous stationary kernel, and k_1 is any continuous integrable real-valued function, which we may choose so that $k_1(0) = 1$ without loss of generality. Here, $\odot$ denotes the Hadamard (or entrywise) product. For any such functions h and k_1, the resulting family, which we emphasise is parametrised by the number of spectral components K, and hyper-parameters $\{\gamma_k, \kappa_k, \omega_k^1, \omega_k^2, U_k, 1 \leq k \leq K\}$, is *general-purpose*.

We proved in the previous chapter that, if $\kappa_k = 0$, U_k is proportional to the identity matrix, and $\omega_k^1 = \omega_k^2$, then the resulting *stationary generalized spectral kernels* (SGSKs)

are pointwise dense in the family of all continuous stationary kernels, and consequently contain elements that may perform as well as any continuous stationary kernel. In order to facilitate the learning of nonstationarity from the data, we introduce a new but equivalent parametrisation of HGSKs. Specifically, we introduce a switch variable $s \in [0, 1]$, and make the changes of variables

$$\kappa_k \to s\kappa_k, \tag{8.3}$$

$$U_k := \begin{bmatrix} a_k & c_k \\ 0 & b_k \end{bmatrix} \to \begin{bmatrix} a_k & sc_k \\ 0 & (1-s)a_k + sb_k \end{bmatrix}, \tag{8.4}$$

$$\omega_k^1 \to \omega_k^m + s\omega_k^d \quad \text{and} \quad \omega_k^2 \to \omega_k^m - s\omega_k^d, \tag{8.5}$$

where

$$\omega_k^m = \frac{1}{2}(\omega_k^1 + \omega_k^2) \in \mathbb{R}^d, \qquad \omega_k^d = \frac{1}{2}(\omega_k^1 - \omega_k^2) \in \mathbb{R}^d.$$

It is easy to see that when $s = 0$, we recover SGSKs, while $s = 1$ corresponds to full GSKs. The parameter s may therefore be used to switch between a family of kernels that may perform as well as any oracle continuous *stationary* kernels, and a family of kernels that may perform as well as any continuous bounded *nonstationary* kernel, so that nonstationarity learning can be thought of as learning $s \in [0, 1]$. This approach is more appealing than setting $s = 1$, although the latter is more general, because placing a prior on s, as we will do later, is tantamount to regularising the off-diagonal mass of the spectral bimeasure of the kernel, which we hope would help mitigating overfitting.

Moreover, for any unbiased (resp. consistent) RFF estimate of h, namely

$$h(x - y) \approx \hat{H}(x)^T \hat{H}(y), \tag{8.6}$$

the kernel

$$\hat{k}_{\text{HGS}}(x, y; K) := \sum_{k=1}^{K} \hat{G}_k(x)^T \hat{G}_k(y), \tag{8.7}$$

with $\hat{G}_k(x) = \hat{H}(x \odot \gamma_k) \otimes (k_1(x \odot \kappa_k)\Psi_k(x))$, where $\otimes$ denotes the Kronecker product, is easily found to be an unbiased (resp. consistent) estimate of k_{HGS}. Both the vanilla RFF expansion of Rahimi and Recht (2007) and the *Fastfood* expansion of Le et al. (2013) may provide such an approximation of h. Thus, HGSKs present the same computational benefits as stationary RFF approaches, with the additional advantage that they allow the learning of nonstationarity from the data.

8.1.3 Should Candidate Kernels be Integrable?

In the previous chapter, we proved that the *general-purposeness* of GSKs is preserved when the inverse input scales γ_k and κ_k are set to 0. We introduced h and k_1 so that the resulting kernels would be integrable, and so that existing vanilla kernels may be regarded as special cases rather than limit cases. We argued that the integrability requirement mitigates overfitting, which might for instance occur when $\gamma_k = \kappa_k = 0$, and when additionally parameters are learned by maximum marginal likelihood, for instance in Gaussian process regression. Indeed, it is well documented that the sparse spectrum kernel, which corresponds to the special case $s = 0$, $\gamma_k = \kappa_k = 0$, is prone to overfitting in Gaussian process regression (see for instance Gal and Turner (2015) and references therein). That being said, the same way we showed that the RFF approximation of the spectral mixture kernel can be regarded as the sparse spectrum kernel with a mixture of Gaussians as prior over the frequency vectors, the same way requiring GSKs to be integrable and using a RFF approximation of h is tantamount to setting $\gamma_k = \kappa_k = 0$ and placing a regularising prior measure over the frequency vectors ω_k^1 and ω_k^2.

In summary, integrability is only needed to avoid overfitting when spectral frequencies ω_k^1 and ω_k^2 are learned directly (without regularisation) as a solution to an optimisation problem such as maximising the marginal likelihood. Alternatively, an adequate prior measure on the spectral frequencies would serve the same purpose in the absence of integrability (i.e. when $\gamma_k = \kappa_k = 0$). In what follows, we will restrict ourselves to the case $\gamma_k = \kappa_k = 0$, and perform adequate Bayesian inference on the spectral frequencies.

8.1.4 Contributions and Outline

In this chapter we propose Bayesian nonparametric approaches to kernel learning with *general-purpose* kernels. Our approaches have training time complexity and training memory requirement that grow in $\mathcal{O}(nd)$, where n is the size of the training data set and d is the dimension of the input space, and with test time complexity and test memory requirement that are independent of n and grow in $\mathcal{O}(d)$. The schemes we propose learns from the data both *model complexity* (K) and whether *nonstationarity* is warranted.

Our approaches consist of considering the *general-purpose* family of kernels of the form

$$k_{\text{NSS}}(x, y; K) = \sum_{k=1}^{K} \Psi_k(x)^T \Psi_k(y), \tag{8.8}$$

where Ψ_k is as per Equation (8.2), with the reparameterisation of Equations (8.4-8.5). We place on s a sigmoid-Gaussian prior

$$s = \sigma(s_x) := \frac{1}{1 + \exp(-s_x)}, \qquad s_x \sim \mathcal{N}(0, 1),$$

which has mode 0.5, so as to be agnostic about whether the kernel should be stationary ($s = 0$) or nonstationary with fully-flexible spectral bimeasure ($s = 1$). We place on $\boldsymbol{\theta}_K := \left\{\theta_k := (\omega_k^m, \omega_k^d, U_k), 1 \leq k \leq K\right\}$ a Dirichlet process (DP) mixture of multivariate Gaussians with *independent coordinates*, conjugate normal-inverse-gamma[2] base distribution, and gamma prior on the concentration parameter. Finally, we place a Poisson prior on the model complexity K, with intensity λ on which we place a conjugate gamma prior. The Poisson prior allows for an arbitrarily large number of spectral components, but only if warranted by the data. The conjugate gamma prior on the intensity of the Poisson helps mitigate the overdispersion of the Poisson distribution. Overall, our prior specification reads

$$\begin{aligned}
s &= \sigma(s_x) && (8.9)\\
s_x &\sim \mathcal{N}(0, 1) && (8.10)\\
\lambda &\sim \Gamma(a_\lambda, b_\lambda) && (8.11)\\
K &\sim \text{Poisson}(\lambda) && (8.12)\\
\alpha &\sim \Gamma(a_\alpha, b_\alpha) && (8.13)\\
\forall i \in [1 \ldots 2d+3],\ G_0[i] &\sim NIG\left(0, \lambda_{G_0}, a_{G_0}, b_{G_0}\right) && (8.14)\\
\forall i \neq j,\ G_0[i] &\perp G_0[j] && (8.15)\\
G &\sim DP(\alpha, G_0) && (8.16)\\
(m_1, v_1), \ldots, (m_K, v_K) | G, K &\sim G && (8.17)\\
\forall k \in [1 \ldots K],\ \theta_k := (\omega_k^m, \omega_k^d, U_k) | m_k, v_k &\sim \mathcal{N}(m_k, \text{diag}(v_k)) && (8.18)\\
\forall i \neq j,\ \theta_i &\perp \theta_j, && (8.19)
\end{aligned}$$

[2]We recall that $(m, v) \sim \text{NIG}(0, \lambda, a, b)$ when $m|v, \lambda \sim \mathcal{N}\left(0, \frac{v}{\lambda}\right)$ and $v^{-1} \sim \Gamma(a, b)$.

where $\text{diag}(v_k)$ is the diagonal matrix with non-negative diagonal v_k. Figure 8.1 illustrates the graphical model corresponding to this prior.

In this Chapter we mainly focus on the three canonical types of machine learning tasks, namely supervised function learning, unsupervised function learning, and semi-supervised function learning, which we solve using kernel methods. For each task, we take as hypothesis space of candidate kernels generalized spectral kernels of the form of Equation (8.8), which we endow with the foregoing prior measure, and we propose a reversible-jump MCMC (RJ-MCMC) scheme (Green (1995)) for Bayesian nonparametric learning of latent functions and/or kernels.

On supervised function learning tasks, our approach, which we elaborate in Section 8.2, improves on BaNK in the following ways:

- It allows the learning from the data of the extent to which stationarity is suitable, which often results in superior accuracy.
- Unlike BaNK, which requires the number of random features m to be fixed and set before inference, our approach allows the learning from the data of the dimension of the feature space (through K), so as to provide a trade-off between scalability and model-fit. This might result in faster inference than BaNK, as the feature space dimension in BaNK is set in relation to the quality of the random Fourier approximation of the spectral mixture kernel, and consequently might be larger than required by the data.
- The independence of the coordinates of the Gaussians in the DP mixture model results in a drastic decrease in training time complexity (resp. memory requirement) from $\mathcal{O}(nmd + md^3)$ (resp. $\mathcal{O}(nmd + md^2)$) to $\mathcal{O}(nmd)$.
- Finally, by treating the concentration parameter as unknown, we may learn the number of mixture components more flexibly.

As for kernel based unsupervised learning, it is covered in Sections 8.3 and 8.6, where we discuss learning kernels in kernel PCA (Schölkopf et al. (1998)) and GP-LVM (Lawrence (2003)) respectively. Kernel based semi-supervised learning is discussed in Section 8.8. Once more, the benefits of our proposed approach are provable flexibility, nonstationarity learning, model complexity learning and scalability.

In addition to these three core types of machine learning tasks, we also propose models for jointly performing manifold and function learning in Section 8.5. Our discussion of joint manifold-function learning provides a more efficient alternative to direct supervised function learning approaches such as BaNK, but more generally GP

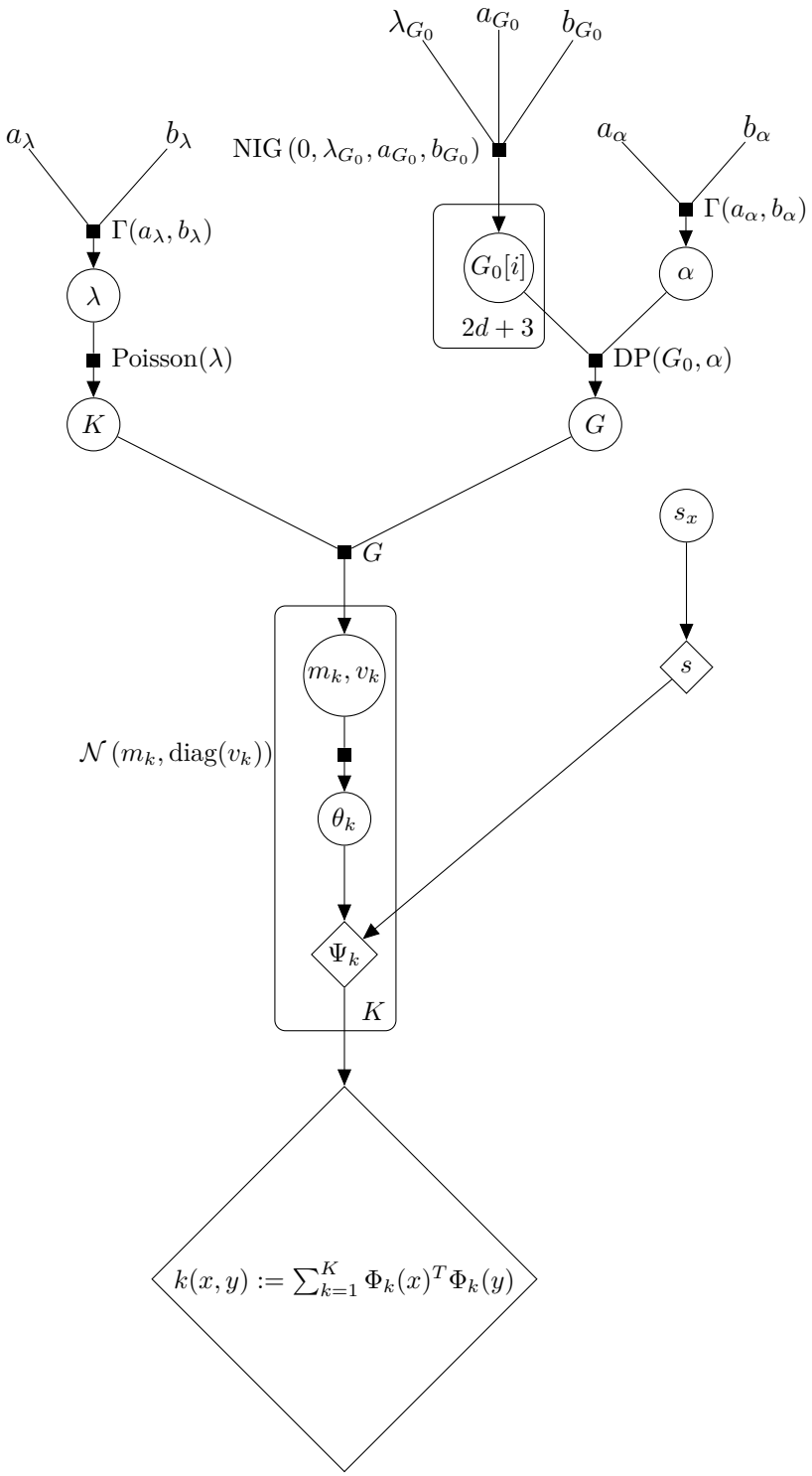

Fig. 8.1 Graphical model of our prior over generalized spectral kernels.

regression and GP classification, when the input space is very large, but the data generating distribution is supported on a lower-dimensional manifold. Autoencoding naturally comes to mind as an example of joint manifold-function learning, and it is indeed within the scope of our proposed approach. More precisely, our approach allows the learning of a distribution over suitable autoencoders. Critically, unlike deep learning based alternatives which require engineering the architectures of the encoding and decoding networks, the complexities of the encoding and decoding functions in our approach are automatically learned from the data in a principled fashion.

Finally, in Section 8.7 we propose a novel kernel based method for learning a generative model for the joint distribution between inputs and labels, namely (x_i, y_i), from an isotropic Gaussian code distribution.

We support our approaches with a wide range of experiments in Section 8.9, and we conclude in Section 8.10.

8.2 Supervised Learning

As previously discussed, the kernels we will consider in this chapter are of the form of k_{NSS} in Equation (8.8). We note that functions that lie in the RKHS induced by k_{NSS} are of the form

$$f(x) = \sum_{k=1}^{K} \beta_k^T \Psi_k(x),$$

where β_k are 4-dimensional real vectors. Thus, whether the kernel method is Bayesian (GP models) or frequentist (RKHS methods), inferring a latent function is equivalent to learning the coefficients β_k. Moreover, when i.i.d. standard normals are placed on the coefficients β_k, the resulting random function is a Gaussian process with mean 0 and covariance function k_{NSS}.

The first class of kernel methods we consider aim at inferring Q latent functions $f_1 \dots f_Q$ from data, and for which we may define a likelihood function that has explicit form $p(\mathcal{D}|f_1, \dots, f_Q, \boldsymbol{u})$ or collapsed form $p(\mathcal{D}|k, \boldsymbol{u})$ for a kernel k, and where $\boldsymbol{u}$ denotes additional likelihood parameters unrelated to the kernel and latent functions. We take k to be of the form k_{NSS} (Equation (8.8)), and we use its RKHS as hypothesis space of candidate latent functions:

$$f_q(x) = \sum_{k=1}^{K} \beta_{kq}^T \Psi_k(x) := \boldsymbol{\beta}_{Kq}^T \boldsymbol{\Xi}_K(x), \tag{8.20}$$

with $\boldsymbol{\beta}_{Kq} = (\beta_{1q}, \dots, \beta_{Kq})$, and $\boldsymbol{\Xi}_K(x) = (\Psi_1(x), \dots, \Psi_K(x))$.

Examples with explicit likelihood include regression and classification problems, while examples with collapsed likelihood include GP regression when i.i.d. standard normal priors are placed on the coordinates of $\boldsymbol{\beta}_{Kq}$, and under a Gaussian noise model. In the explicit case, the uncertainties are the choice of hypothesis space of candidate functions (i.e. the RKHS or equivalently its reproducing kernel) and the choice of functions f_q in the chosen hypothesis space that are suitable for the task of interest, whereas in the collapsed case the only uncertainty is the choice of kernel, conditional on which the latent functions are known.

Our goal is to place a prior on the kernel k (or equivalently on K, $\boldsymbol{\theta}_K$ and s) and $\boldsymbol{u}$, and to make Bayesian inference on the latent functions f_q (or equivalently on $\boldsymbol{\beta}_K := (\boldsymbol{\beta}_{K1}, \ldots, \boldsymbol{\beta}_{KQ})$). To do so in the collapsed case, considering that the optimal solutions (i.e. $E(f_1, \ldots, f_Q|\mathcal{D}, k, \boldsymbol{u})$ for GP regression) are readily available from the kernel and $\boldsymbol{u}$, it suffices to sample from the posterior $p(K, \boldsymbol{\theta}_K, s, \boldsymbol{u}|\mathcal{D})$. In the explicit case, we additionally need to place a prior on $\boldsymbol{\beta}_K$ so as to sample from $p(\boldsymbol{\beta}_K, K, \boldsymbol{\theta}_K, s, \boldsymbol{u}|\mathcal{D})$. In both cases, considering that the dimension of $\boldsymbol{\theta}_K$ depends on K, our problem is inherently cross-dimensional, and we will use RJ-MCMC techniques (Green (1995)) to sample from the posterior.

For problems such as GP regression that admit both a collapsed and an explicit likelihood form, the collapsed representation will typically lead to a sampler that converges faster in terms of number of MCMC iterations, however both test time complexity and test memory requirement will depend on the size of the training sample, as predictions will likely require estimating

$$\mathrm{E}\left(f_1, \ldots, f_Q|\mathcal{D}, K_i, \boldsymbol{\theta}_{K_i}, s_i, \boldsymbol{u}_i\right)$$

where $(K_i, \boldsymbol{\theta}_{K_i}, s_i, \boldsymbol{u}_i) \sim p(K, \boldsymbol{\theta}_K, s, \boldsymbol{u}|\mathcal{D})$, using states of the RJ-MCMC sampler. The explicit representation on the other hand might lead to slightly slower (in terms of number of MCMC iterations) mixing of the RJ-MCMC sampler, as the state of the Markov chain includes more variables, but test time complexity and memory requirement would be independent of the size of the training dataset, as predictions merely require storing a collection of samples $(\boldsymbol{\beta}_i, \boldsymbol{\theta}_{K_i}, s_i)$ from the sampler post-burn-in. Which representation should be preferred depends on the amount of resources available at prediction time.

8.2.1 Prior Specification

For convenience, we place a multivariate standard normal prior on $\boldsymbol{\beta}_K$, namely

$$\boldsymbol{\beta}_K \sim \mathcal{N}(0, I),$$

which is equivalent to placing independent Gaussian process priors on the latent functions

$$f_q \overset{\text{i.i.d.}}{\sim} \mathcal{GP}(0, k_{\text{NSS}}).$$

Extensions to other priors pose no additional challenge. When the likelihood has additional parameters $\boldsymbol{u}$, we choose the prior on $p(\boldsymbol{u})$ to be conjugate to the likelihood when possible, for instance an inverse-gamma prior on the noise variance in GP regression. The prior specification on the kernel was discussed in Section 8.1.4, and is illustrated in Figure 8.1. We recall that it reads

$$\begin{aligned}
s &= \sigma(s_x) && (8.21)\\
s_x &\sim \mathcal{N}(0,1) && (8.22)\\
\lambda &\sim \Gamma(a_\lambda, b_\lambda) && (8.23)\\
K &\sim \text{Poisson}(\lambda) && (8.24)\\
\alpha &\sim \Gamma(a_\alpha, b_\alpha) && (8.25)\\
\forall i \in [1 \dots 2d+3],\ G_0[i] &\sim NIG\left(0, \lambda_{G_0}, a_{G_0}, b_{G_0}\right) && (8.26)\\
\forall i \neq j,\ G_0[i] &\perp G_0[j] && (8.27)\\
G &\sim DP(\alpha, G_0) && (8.28)\\
(m_1, v_1), \dots, (m_K, v_K) | G, K &\sim G && (8.29)\\
\forall k \in [1 \dots K],\ \theta_k := (\omega_k^m, \omega_k^d, U_k) | m_k, v_k &\sim \mathcal{N}(m_k, \text{diag}(v_k)) && (8.30)\\
\forall i \neq j,\ \theta_i &\perp \theta_j. && (8.31)
\end{aligned}$$

The conjugate Poisson-gamma prior we place on K, enables easily specifying the prior mean and prior variance over K independently of each other. In effect, by the laws of total expectation and total variance, the prior mean and prior variance of K are found to be

$$\text{E}(K) = \frac{a_\lambda}{b_\lambda} \quad \text{and} \quad \text{Var}(K) = \frac{a_\lambda}{b_\lambda}\left(1 + \frac{1}{b_\lambda}\right). \tag{8.32}$$

These equations allow us to choose a_λ and b_λ so as to express prior belief about average model complexity (for instance setting $\text{E}(K)$ to a small fraction of training dataset size n), while remaining relatively uninformative by setting a high enough prior variance

$\mathrm{Var}(K)$. We choose remaining parameters $\lambda_{G_0}, a_{G_0}, b_{G_0}, a_\alpha, b_\alpha$ in a similar spirit, setting the mean of the variable they drive to a reasonable guess, and choosing its variance to be large enough to express uninformativeness. Figure 8.2 illustrates the graphical model of supervised learning with general-purpose kernels when the likelihood is i.i.d.

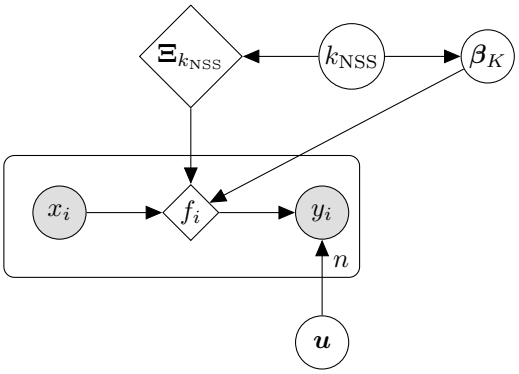

Fig. 8.2 Graphical model for direct and explicit supervised learning with GSKs under i.i.d. likelihoods. The full graphical model of our prior over the kernel k_{NSS} is depicted in Figure 8.1.

8.2.2 Reversible-Jump MCMC Sampler

As K is to be inferred, the number of state variables in our prior specification is not constant, and consequently vanilla MCMC techniques do not apply. On the one hand, if K was fixed, we would have used standard MCMC techniques, together with standard Gibbs sampling techniques for sampling from posteriors under Dirichlet process priors (as discussed for instance in Escobar and West (1995) and Neal (2000)). The resulting inference scheme would have been very similar to the BaNK approach proposed by Oliva et al. (2016). On the other hand, under our Poisson prior on K, had we placed as prior over the spectral hyper-parameters θ_k that they are independent draws from a *finite* mixture of Gaussians, we could have written down the conditional (for a given K) posterior probability density function (pdf) as

$$p\left(\boldsymbol{\theta}_K, s, \boldsymbol{\beta}_K, \boldsymbol{u}, m_1, v_1, \ldots, m_K, v_K | \mathcal{D}, K\right) \propto p\left(\mathcal{D} | \boldsymbol{\theta}_K, s, \boldsymbol{\beta}_K, \boldsymbol{u}\right) p(\boldsymbol{\beta}_K) p(\boldsymbol{u}) p(s) \prod_{k=1}^{K} p(\theta_k | \bar{m}_{c_k}, \bar{v}_{c_k}) \prod_{k=1}^{K_c} p(\bar{m}_k, \bar{v}_k), \quad (8.33)$$

where $\bar{m}_k, \mathrm{diag}\left(\bar{v}_k\right)$ are means and covariance matrices of the K_c distinct clusters. Our problem would then have been a standard cross-dimensional MCMC problem, in other words a Bayesian model selection problem under a countable family of models indexed

by $K \in \mathbb{N}$, each of which has a fixed number of parameters, and has a joint distribution that admits a pdf given by Equation (8.33). This would have been the standard setup for which reversible-jump MCMC (RJ-MCMC) was proposed by Green (1995).

However, under our prior specification, standard RJ-MCMC does not apply as our Dirichlet process prior over (m_k, v_k) is neither a discrete distribution, nor does it admit a pdf. The challenge here is therefore to extend the work of Green (1995) to the case where the variables whose numbers change between models, say x_k (here x_k would be $(\beta_{kq}, \theta_k, m_k, v_k)$), have a prior structure that is a stochastic process whose marginals do not necessarily admit a pmf (probability mass function) or a pdf, but under which we may still sample from the conditional $x_{K+1}|x_1, \ldots, x_K$, where $(x_1, \ldots, x_K, x_{K+1})$ are distributed as per the prior process. A Dirichlet process mixture of Gaussians is an example of such prior stochastic process.

Notwithstanding the aforementioned challenge, to derive a Markov transition kernel that leaves the posterior distribution

$$\boldsymbol{\theta}_K, s, \boldsymbol{\beta}_K, \boldsymbol{u}, m_1, v_1, \ldots, m_K, v_K, \alpha, \lambda, K \mid \mathcal{D}$$

invariant, it suffices to proceed in a Gibbs sampling fashion, and in each Gibbs cycle perform three types of updates. The first type of update, which, following the standard RJ-MCMC jargon, we denote *within-model update*, leaves the distribution

$$\boldsymbol{\theta}_K, s, \boldsymbol{\beta}_K, \boldsymbol{u}, \lambda \mid \mathcal{D}, m_1, v_1, \ldots, m_K, v_K, \alpha, K$$

invariant. This type of update does not present any theoretical challenge, as the above distribution admits a pdf, and the update can be constructed using standard MCMC techniques (Brooks et al. (2011)). The second type of update, which we will refer to as *Dirichlet process update*, leaves the distribution

$$m_1, v_1, \ldots, m_K, v_K, \alpha \mid \mathcal{D}, K, \boldsymbol{\theta}_K, s, \boldsymbol{\beta}_K, \boldsymbol{u}$$

(which is the same as $m_1, v_1, \ldots, m_K, v_K, \alpha \mid K, \boldsymbol{\theta}_K$) invariant. This type of update is analogous to a Gibbs cycle when sampling from the posterior in a Dirichlet process mixture of Gaussians model, and the techniques discussed in Escobar and West (1995) and Neal (2000) can be readily used. Finally, the third type of update, which, using the RJ-MCMC jargon once more, we refer to as *between-models update*, leaves the distribution

$$\boldsymbol{\beta}_K, K, \boldsymbol{\theta}_K, m_1, v_1, \ldots, m_K, v_K \mid \mathcal{D}, s, \boldsymbol{u}, \alpha, \lambda$$

invariant. It is indeed easy to see that applying all three updates in a Gibbs cycle will leave the full posterior distribution invariant. Finally, for our RJ-MCMC sampler to be valid, that is to satisfy the conditions of Birkhoff's ergodic theorem (Cornfeld et al. (2012)) so that we may estimate expectations of functions under the posterior, it is sufficient that the Markov transition kernel of a full Gibbs cycle be aperiodic and π-irreducible, where we use π here to denote the posterior probability measure.[3] If our Markov transition kernel is periodic, then the transition kernel of the within-model update (as an operator on within-model variables) has to be periodic. Consequently, considering that we will use standard MCMC techniques for the within-model update,[4] the Markov transition kernel of a full Gibbs cycle will be aperiodic. In regards to irreducibility, it is sufficient that the transition kernels of all three types of update (as operators on variables that may change in the corresponding update) be irreducible. This will be satisfied by within-model and Dirichlet process update types, as the familiar techniques we will use guarantee irreducibility. As for the between-models update, it suffices that it be irreducible relative to the only variable that does not change in any of the two other types of update, namely K.

In summary, to fully tackle the challenge inherent to allowing K to vary in our generative model, we need to devise a Markov transition kernel for the between-models update scheme that i) leaves the distribution

$$\boldsymbol{\beta}_K, K, \boldsymbol{\theta}_K, m_1, v_1, \ldots, m_K, v_K \mid \mathcal{D}, s, \boldsymbol{u}, \alpha, \lambda$$

invariant for every $s, \boldsymbol{u}, \alpha, \lambda$, and that ii) is irreducible (as an operator on the variable K). In the following section, we propose such a between-models Markov transition kernel, which we prove satisfies the *detailed-balance* condition relative to the distribution

$$\boldsymbol{\beta}_K, K, \boldsymbol{\theta}_K, m_1, v_1, \ldots, m_K, v_K \mid \mathcal{D}, s, \boldsymbol{u}, \alpha, \lambda,$$

and consequently leaves this foregoing distribution invariant,[5] and which we verify is irreducible with regards to K.

[3] We recall that a Markov transition kernel $P(x, dx')$ operating on an input space $\mathcal{C}$ is π-irreducible if for every $A \subseteq \mathcal{C}$ such that $\pi(A) > 0$ and for every $x \in \mathcal{C}$, we have $P^n(x, A) := \int_A P^n(x, dx') > 0$ for some $n \in \mathbb{N}$. In other words, the transition kernel can take us into any set of non-null probability from anywhere after a sufficiently large (but finite) number of steps.

[4] We recall that these techniques give rise to aperiodic and irreducible Markov chains.

[5] We recall that a Markov transition kernel $P(x, dx')$ satisfies the detailed-balance condition with respect to a probability measure $\pi(dx)$ when

$$\forall A, B \subseteq \mathcal{C}, \quad \int_A \int_B P(x, dx')\pi(dx) = \int_B \int_A P(x', dx)\pi(dx'),$$

Between-Models Update

The intuition behind our approach to constructing our between-models update is best found in the case where our Dirichlet process mixture prior, which is effectively an *infinite* mixture of Gaussians, is replaced with i.i.d. draws from a *finite* mixture of Gaussians. In that case, standard RJ-MCMC techniques apply. Moreover, if our Markov transition kernel is a Hastings kernel that consists of moves to increase K by sampling a number of additional sets of variables (θ_k, m_k, v_k) from the prior conditional on the existing variables $(\boldsymbol{\theta}_K, m_1, v_1, \ldots, m_K, v_K, \alpha)$ and by sampling the same number of new coordinate vectors β_{kq} from a proposal with density $\hat{p}(\beta_{kq})$, and moves to decrease K by deleting a number of existing sets of variables (β_{kq}, θ_k), then by applying the seminal result of Green (1995), an acceptance probability sufficient to preserve detailed-balance is found to be

$$r_{K\to K^*} = \min\left(1, \frac{p\left(\mathcal{D}|\boldsymbol{\theta}^*_{K^*}, s, \boldsymbol{\beta}^*_{K^*}, \boldsymbol{u}\right)}{p\left(\mathcal{D}|\boldsymbol{\theta}_K, s, \boldsymbol{\beta}_K, \boldsymbol{u}\right)} \frac{p(K^*)}{p(K)} \frac{j_{K^*}(K)}{j_K(K^*)} \prod_{k\in\mathcal{K},\ q\in[1\ldots Q]} \frac{p(\beta_{kq})}{\hat{p}(\beta_{kq})}\right), \tag{8.34}$$

when $K^* > K$, and

$$r_{K\to K^*} = \min\left(1, \frac{p\left(\mathcal{D}|\boldsymbol{\theta}^*_{K^*}, s, \boldsymbol{\beta}^*_{K^*}, \boldsymbol{u}\right)}{p\left(\mathcal{D}|\boldsymbol{\theta}_K, s, \boldsymbol{\beta}_K, \boldsymbol{u}\right)} \frac{p(K^*)}{p(K)} \frac{j_{K^*}(K)}{j_K(K^*)} \prod_{k\in\mathcal{K},\ q\in[1\ldots Q]} \frac{\hat{p}(\beta_{kq})}{p(\beta_{kq})}\right), \tag{8.35}$$

when $K^* < K$, where K^* is the proposed new number of spectral components, $\mathcal{K}$ is the set of indices of variables that have either been added or deleted to construct the proposal, $j_K(K^*)$ is the probability of attempting to change the number of spectral components from K to K^*, and $\boldsymbol{\beta}^*_{K^*}$ and $\boldsymbol{\theta}^*_{K^*}$ are the proposed new coordinates and hyper-parameters. Interestingly, this result does not depend on the number of Gaussian mixture components in the prior, and we would hope that it still holds in the infinite limit that is our Dirichlet process mixture prior, although we would like to stress that our setup is in fact not covered by the seminal result of Green (1995) because, in the Dirichlet process limit, the proposal $\theta_{K+1} \mid \boldsymbol{\theta}_K, m_1, v_1, \ldots, m_K, v_K, \alpha$ no longer admits a pdf or a pmf. That being said, under our Dirichlet process mixture prior, for every

where $\mathcal{C}$ is the full input space. To show that P satisfies detailed-balance with respect to π implies that π is invariant by P, it suffices to take $A = \mathcal{C}$. We then get

$$\forall B \subseteq \mathcal{C}, \quad (P \circ \pi)(B) := \int_{\mathcal{C}} \int_B P(x, dx')\pi(dx) = \int_B \int_{\mathcal{C}} P(x', dx)\pi(dx') = \int_B \pi(dx') := \pi(B).$$

K, we may still sample from the conditional

$$\beta_{(K+1)q}, \theta_{K+1}, m_{K+1}, v_{K+1} \mid \boldsymbol{\beta}_K, \boldsymbol{\theta}_K, m_1, v_1, \dots m_K, v_K, \alpha,$$

and it turns out that the acceptance probability of Equations (8.34, 8.35) indeed remains a sufficient condition for detailed-balance of the aforementioned Hastings kernel relative to

$$\boldsymbol{\beta}_K, K, \boldsymbol{\theta}_K, m_1, v_1, \dots, m_K, v_K \mid \mathcal{D}, s, \boldsymbol{u}, \alpha, \lambda.$$

Moreover, the resulting Hastings transition kernel is indeed irreducible with respect to K. These results are formalised in Proposition E.1, and proved in Section E.1.

In our experiments, we will typically restrict ourselves to attempts to increment or decrement K by 1, but other alternatives present no practical or theoretical challenge. We choose $j_K(.)$ such that from $K = 0$ we may only add one spectral component ($j_0(1) = 1$) and such that for $K > 0$ we attempt to increase and decrease K with equal probability ($\forall K > 0,\ j_K(K+1) = j_K(K-1) = 0.5$). By convention, when $K = 0$, the kernel and all latent functions are equal to 0. When we consider adding one new spectral component, we may sample from the proposal θ_{K+1} using standard results on the Dirichlet process (see for instance Equation (6) in Teh (2011)), which we recall below:

$$(m_{K+1}, v_{K+1}) \mid m_1, v_1, \dots m_K, v_K, \alpha \sim \frac{1}{\alpha + K}\left(\alpha G_0 + \sum_{k=1}^{K} \delta_{(m_k, v_k)}\right) \tag{8.36}$$

and

$$\theta_{K+1} \mid m_{K+1}, v_{K+1} \sim \mathcal{N}\left(m_{K+1}, v_{K+1}\right). \tag{8.37}$$

We choose as proposal on β_{K+1} its (multivariate standard Gaussian) prior.

Within-Model Updates

The conditional posterior pdf of s_x given everything else is proportional to

$$p\left(\mathcal{D} | \theta_1, \dots, \theta_K, \sigma(s_x), \boldsymbol{\beta}_K, \boldsymbol{u}\right) \mathcal{N}(s_x | 0, 1), \tag{8.38}$$

where we use $\mathcal{N}(.|m, \Sigma)$ to denote the pdf of a Gaussian random variable with mean m and variance or covariance matrix Σ. Similarly, the conditional posterior pdf of

$\theta_1, \ldots, \theta_K$ given everything else, including $m_1, v_1, \ldots, m_K, v_K$, is proportional to

$$p\left(\mathcal{D}|\theta_1, \ldots, \theta_K, \sigma(s_x), \boldsymbol{\beta}_K, \boldsymbol{u}\right) \prod_{k=1}^{K} \mathcal{N}\left(\theta_k|m_k, \mathrm{diag}(v_k)\right). \tag{8.39}$$

As for the conditional posterior pdf of $\boldsymbol{\beta}_K$ given everything else, it is proportional to

$$p\left(\mathcal{D}|\theta_1, \ldots, \theta_K, \sigma(s_x), \boldsymbol{\beta}_K, \boldsymbol{u}\right) \mathcal{N}(\boldsymbol{\beta}_K|0, I). \tag{8.40}$$

Considering the form of the conditional posteriors of Equations (8.38 - 8.40), we use the Elliptical Slice Sampling (ESS) algorithm (Murray et al. (2010)) for the within-model update of s_x, $\boldsymbol{\beta}_K$, and $\theta_1, \ldots, \theta_K$, which we prefer to the Metropolis-Hastings algorithm (MH) as it does not require tuning.

When $p(\boldsymbol{u})$ is conjugate to the likelihood, we update $\boldsymbol{u}$ by sampling directly from the posterior $p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\theta}_K, s, \boldsymbol{\beta}_K)$. Alternatively, we perform a MH update, drawing a proposal $\boldsymbol{u}^*$ from the prior, which we accept with probability

$$r_{\boldsymbol{u}\to\boldsymbol{u}^*} = \min\left(1, \frac{p\left(\mathcal{D}|\theta_1, \ldots, \theta_K, s, \boldsymbol{\beta}_K, \boldsymbol{u}^*\right)}{p\left(\mathcal{D}|\theta_1, \ldots, \theta_K, s, \boldsymbol{\beta}_K, \boldsymbol{u}\right)}\right). \tag{8.41}$$

We update λ by sampling from $p(\lambda|K)$, which by conjugacy of the Gamma prior to the Poisson likelihood is also Gamma and reads

$$p(\lambda|K) = \gamma(\lambda; a_\lambda + K, b_\lambda + 1). \tag{8.42}$$

Dirichlet Process Update

It is easy to see from our prior specification that

$$m_1, v_1, \ldots, m_K, v_K, \alpha \mid \mathcal{D}, K, \theta_1, \ldots, \theta_K, s, \boldsymbol{\beta}_K, \boldsymbol{u}$$

is the same as

$$m_1, v_1, \ldots, m_K, v_K, \alpha \mid K, \theta_1, \ldots, \theta_K.$$

Consequently, any ergodic update that leaves $m_1, v_1, \ldots, m_K, v_K, \alpha \mid K, \theta_1, \ldots, \theta_K$ invariant, which includes the approaches discussed in Sections 3 and 7 of Neal (2000) and in Sections 2 and 6 of Escobar and West (1995), may serve as Dirichlet process update. The standard Gibbs sampling approach, which we adopt here, proceeds as

follows. Cluster assignment variables c_k are introduced such that

$$m_k = \bar{m}_{c_k}, \qquad v_k = \bar{v}_{c_k},$$

where the set of cluster parameters $\{\bar{m}_1, \bar{v}_1, \ldots, \bar{m}_{K_c}, \bar{v}_{K_c}\}$ is the same as the set $\{m_1, v_1, \ldots, m_K, v_K\}$, but without duplicates, and where K_c is the number of unique clusters to which at least one sample θ_k is assigned. The approach then consists of sequentially updating each cluster assignment c_k conditional on other cluster assignments, samples $\theta_1, \ldots, \theta_K$, cluster parameters $\bar{m}_1, \bar{v}_1, \ldots, \bar{m}_{K_c}, \bar{v}_{K_c}$, and the concentration parameter α, then updating cluster parameters conditional on cluster assignments and samples, and finally updating the concentration parameter α conditional on the number of cluster K_c.

Denoting c_{-k} the set of all cluster assignments but the k-th, and by $K_{-k,i}$ the number of samples associated to the i-th cluster excluding θ_k, the posterior probability of the cluster to which θ_k belongs is given by

$$\mathbb{P}(c_k = i \mid \theta_k, c_{-k}, \bar{m}_i, \bar{v}_i, \alpha, \ldots) = b \frac{K_{-k,i}}{K - 1 + \alpha} \mathcal{N}\left(\theta_k | \bar{m}_i, \mathrm{diag}(\bar{v}_i)\right) \tag{8.43}$$

when $i \in [1 \ldots K_c]$ is an existing cluster, and

$$\mathbb{P}(c_k \notin [1 \ldots K_c] \mid \theta_k, \alpha, \ldots) = b \frac{\alpha}{K - 1 + \alpha} \prod_{j=1}^{2d+3} \int \mathcal{N}\left(\theta_k[j] \mid m_j, v_j\right) G_0[j](dm_j, dv_j), \tag{8.44}$$

where b is a normalising constant. By conjugacy of the normal-inverse-gamma prior to the Gaussian likelihood the marginal likelihood

$$\int \mathcal{N}\left(\theta_k[j] \mid m_j, v_j\right) G_0[j](dm_j, dv_j)$$

is available in closed form (see for instance Section 6 of Bishop (2007a)). When it is necessary to add a new cluster, its parameters (m, v) are drawn from the base distribution G_0, and empty clusters are forgotten.

Conditional on the samples that have been assigned to cluster k, the posterior distribution of $(\bar{m}_k, \bar{v}_k)$ is easily found to exhibit no cross-dependency between the $2d+3$ coordinates representing the parameters ω_k^1, ω_k^2 and U_k. Moreover, for every cluster i and coordinate j, the posterior distribution of $(\bar{m}_i[j], \bar{v}_i[j])$ given all $\theta_k[j]$ such that $c_k = i$, is identical to the NIG posterior predictive distribution in a (conjugate) Bayesian model with i.i.d. (scalar) Gaussian likelihood and conjugate NIG $(0, \lambda_{G_0}, a_{G_0}, b_{G_0})$ prior

on the mean and variance. We refer the reader to Section 6 of Bishop (2007a) for explicit formulae.

Finally, the posterior distribution over the concentration parameter was found by Escobar and West (1995) (Section 6) to be the a mixture of gamma distributions. More precisely, we sample a variable η from the following beta distribution conditional on α

$$\eta \mid \alpha, K_c \sim \text{Beta}(\alpha + 1, K_c),$$

and then sample α conditional on η from the following mixture of gamma distributions:

$$\alpha \mid \eta, K_c \sim \pi_\eta \gamma\left(a_\alpha + K_c, b_\alpha - \log \eta\right) + (1 - \pi_\eta)\gamma(a_\alpha + K_c - 1, b_\alpha - \log \eta), \quad (8.45)$$

where

$$\frac{\pi_\eta}{1 - \pi_\eta} = \frac{a_\alpha + K_c - 1}{K(b_\alpha - \log \eta)}.$$

To summarise, the Dirichlet process update is very similar to the standard Gibbs sampling approach to learning a Dirichlet process mixture of (scalar) Gaussians under a normal-inverse-gamma base distribution, and, in the implementation, this step can be isolated into a separate self-contained density estimation module that can be tested individually. Crucially, thanks to the independence between the coordinates of G_0, at no point during an update do we need to sample from a fully-dependent d-dimensional random variable, which helps preserve a time complexity linear in d.

Prediction

Samples from the posterior $p(f_1, \ldots, f_Q|\mathcal{D})$ are easily formed from samples $(\boldsymbol{\beta}_{K_i}, \boldsymbol{\theta}_{K_i}, s_i)$ (post-burn-in) using Equations (8.20) and (8.8). As it is standard practice, sample mean and sample variance may then be used as consistent estimates of posterior mean and posterior variance (by Birkhoff's ergodicity theorem).

Computational Cost

At training time, the computational bottleneck of within-model and between-models updates is the evaluation of the model likelihood, which will typically have time complexity (resp. memory requirement) linear in n, d, Q, and K, more precisely $\mathcal{O}(ndK + nKQ)$ (resp. $\mathcal{O}(nd + nK + KQ)$), in the case of i.i.d. likelihoods such as regression and classification. As for Dirichlet process updates, they have time complexity and memory requirement independent of n, and linear in K and d.

At test time, resource requirements are independent of n in the explicit case, and grow linearly with the input dimension d. In the case of collapsed likelihoods, resource requirements are not worse than at training time.

8.3 Kernel PCA

The second class of kernel methods we consider is non-linear dimensionality reduction through kernel PCA (Schölkopf et al. (1998)).

Kernel PCA is the non-linear generalisation of PCA introduced by Schölkopf et al. (1998), consisting of performing Principal Component Analysis in the feature space of a kernel k_e. More precisely, we recall that PCA aims at finding *orthonormal* directions w_i in the input space in which the data vary the most, namely,

$$\begin{cases} w_0 = \arg\max_{w} \frac{1}{n} \sum_{i=1}^{n} \left(x_i^T w\right)^2 \\ w_j = \arg\max_{w \perp w_1, \dots, w_{j-1}} \frac{1}{n} \sum_{i=1}^{n} \left(x_i^T w\right)^2, \quad \forall j > 0. \end{cases} \tag{8.46}$$

A solution to this problem is given by an orthonormal system of eigenvectors of the empirical covariance matrice $\boldsymbol{S}$,

$$\boldsymbol{S} = \frac{1}{n} \sum_{i=1}^{n} (x_i - \bar{m})(x_i - \bar{m})^T, \qquad \text{with} \qquad \bar{m} = \frac{1}{n} \sum_{i=1}^{n} x_i,$$

sorted by decreasing order of eigenvalue.

Schölkopf et al. (1998) generalised PCA by, instead, seeking eigenvectors of the covariance matrix of features

$$\boldsymbol{S}_{k_e} = \frac{1}{n} \sum_{i=1}^{n} \left(\boldsymbol{\Xi}_{k_e}(x_i) - \bar{m}_{k_e}\right) \left(\boldsymbol{\Xi}_{k_e}(x_i) - \bar{m}_{k_e}\right)^T, \tag{8.47}$$

with

$$\bar{m}_{k_e} = \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{\Xi}_{k_e}(x_i),$$

where $\boldsymbol{\Xi}_{k_e}$ is a feature map of the kernel k_e, for instance the one whose existence is guaranteed by Mercer's theorem, so that

$$k_e(x, y) = \langle \boldsymbol{\Xi}_{k_e}(x), \boldsymbol{\Xi}_{k_e}(y) \rangle_{L^2}.$$

Strictly speaking, when the feature space (i.e. the induced RKHS) $\mathcal{H}_{k_e}$ is infinite dimensional, $\boldsymbol{S}_{k_e}$ is no longer a matrix; it is the linear operator that maps a function $f \in \mathcal{H}_{k_e}$ to the function

$$\boldsymbol{S}_{k_e}[f] := \frac{1}{n}\sum_{i=1}^{n} (\boldsymbol{\Xi}_{k_e}(x_i) - \bar{m}_{k_e}) \, \langle \boldsymbol{\Xi}_{k_e}(x_i) - \bar{m}_{k_e}, f \rangle_{\mathcal{H}_{k_e}},$$

but we may still define eigenvectors as functions $f \in \mathcal{H}_{k_e}$ satisfying

$$f \neq 0, \qquad \boldsymbol{S}_{k_e}[f] = \lambda f, \qquad \text{for some} \qquad \lambda \in \mathbb{R}.$$

The genius in the work of Schölkopf et al. (1998) is that the authors demonstrated that, even when the feature space is infinite-dimensional, eigenvectors of the linear operator $\boldsymbol{S}_{k_e}$ are linear combinations of $\boldsymbol{\Xi}_{k_e}(x_i)$, and the coefficients of the linear combination are obtained as eigenvectors of the Gram matrix $(k_e(x_i, x_j))_{1 \leq i,j \leq n}$, which, crucially, does not require evaluating the feature map $\boldsymbol{\Xi}_{k_e}$. For kernels inducing infinite-dimensional RKHSs, such as the RBF kernel, this so-called *kernel trick* is imperative. However, when care is taken to learn the dimension of the RKHS as a model complexity parameter, which we have been arguing for thus far, we may directly work in the feature space. In this case, $\boldsymbol{S}_{k_e}$ is a positive semi-definite matrix, and its eigenvectors are easily obtained by SVD.

We note that PCA is recovered as the special case where k_e is the linear kernel. In general though, kernel PCA improves on linear PCA for dimensionality reduction in that, when the kernel is properly chosen, for a given number of principal components or compression budget c, kernel PCA may capture more information about the data generating distribution than linear PCA.[6] The approach commonly used to learn k_e consists of introducing a pseudo pre-image

$$\hat{x}_{k_e}(x) = \underset{x' \in \mathbb{R}^d}{\arg\min} \, ||\boldsymbol{\Xi}_{k_e}(x') - P_c \boldsymbol{\Xi}_{k_e}(x)||^2, \tag{8.48}$$

where $P_c \boldsymbol{\Xi}_{k_e}(x)$ denotes the projection of $\boldsymbol{\Xi}_{k_e}(x)$ onto the c principal components,[7] and to choose k_e so as to minimise the normalized mean squared reconstruction error,

[6] For data visualisation purposes for instance, one will typically choose $c = 2$, and represent an input point x by the coordinates of the projection of its associated feature $\boldsymbol{\Xi}_{k_e}(x)$ onto the basis formed by the 2 principal components.

[7] The feature vectors here are assumed centred.

namely

$$E(k) = \sqrt{\frac{\sum_{i=1}^{n} ||\hat{x}_k(x_i) - x_i||^2}{\sum_{i=1}^{n} ||x_i||^2}},$$

and

$$k_e = \arg\min_{k \in \mathcal{K}} E^2(k), \tag{8.49}$$

where $\mathcal{K}$ is some hypothesis space of candidate kernels.

8.3.1 Model Specification

Our goal in this Section is to extend this approach to allow for general-purpose Bayesian nonparametric kernel learning. For every input datum x_i, we introduce a *synthetic observation* we refer to as *intrinsic reconstruction error* and which we denote e_i, that can be viewed as the minimum reconstruction error one would expect from doing kernel PCA with an *oracle kernel* and determining the pseudo pre-image of x_i under a given compression budget. We use as generative model for e_i that they are i.i.d. Gaussians with mean the root mean squared reconstruction error $E(k)$, namely

$$e_i \mid k \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(E(k), \sigma^2\right). \tag{8.50}$$

Clearly, when one would hope for a perfect reconstruction, one could set the synthetic observations e_i to 0, and the maximum likelihood solution would coincide with the standard approach to kernel learning in PCA (Equation (8.49)), irrespective of the value of σ^2. Consequently, we will take $\sigma^2 = 1$ from now on, without loss of generality. To perform full Bayesian inference, we need to place a prior on the kernel k, which we take to be of the form previously introduced (Equations (8.21 - 8.31)). Figure 8.3 illustrates the graphical model for our approach.

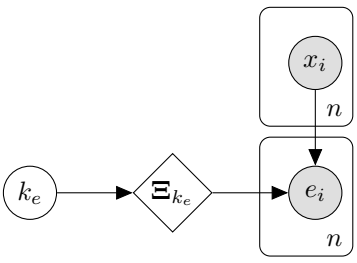

Fig. 8.3 Graphical model for Bayesian nonparametric kernel PCA with general-purpose kernels.

Remark 8.1 *In general, the pseudo pre-image $\hat{x}_k(x)$ will not be available in closed form, and one will have to resort to gradient-based techniques to solve the Problem (8.48). Strictly speaking, we mean for $E(k)$ in the generative model Equation (8.50) to be the output of the deterministic optimiser used to solved Problem (8.48), rather than a solution to the optimisation problem itself. So long as the optimiser used to solve Problem (8.48) is the same as one would use to reconstruct an input after dimensionality reduction, we believe this is a fair assumption to make: $E(k)$ accounts for the root mean square reconstruction error we would obtain using our best tool available.*

8.3.2 Inference

We use synthetic observations $c_i = 0$ to express that we hope for perfect reconstruction, and are interested in sampling from the posterior distribution over the kernel

$$k \mid c_1, \ldots, c_n.$$

RJ-MCMC Updates

This problem turns out to be exactly the same as collapsed supervised learning with $\mathcal{D} = (c_1, \ldots, c_n)$, for which we already proposed an RJ-MCMC sampling scheme.

Computational Cost

Although the computational cost is still linear in the number of input points n, the $\mathcal{O}$ constant will typically be considerably larger than in regression and classification problems, due to the need to solve n optimisation problems to compute a single likelihood term

$$p(\mathcal{D} \mid k) = \frac{1}{(2\pi)^{\frac{1}{2}}} e^{-\frac{1}{2}\sum_{i=1}^{n} ||\hat{x}_k(x_i) - x_i||^2}.$$

To speed up convergence of optimisations, when the likelihood is evaluated at a proposal kernel k^* that is only marginally different from the current kernel k, we initialise the optimisers to the pseudo pre-image corresponding to kernel k. When distributed computing is available, we might also solve the foregoing optimisation problems in parallel. The computational issue here is not inherent to our approach, it is a consequence of the need to compute all pseudo pre-images to evaluate the quality of dimensionality reduction for a given kernel. That being said, for some kernels such as the RBF kernel, closed-form formulae for the pseudo pre-image do exist, but such kernels are often too simplistic.

The two most popular usages of kernel PCA are perhaps data visualisation, and pre-processing for supervised learning. In following sections, we propose direct approaches to perform these two tasks that are more computationally efficient than full Bayesian nonparametric kernel PCA. The idea of encoding as 'pre-processing' underpins the contribution of Section 8.5 where we discuss simultaneously learning a manifold on which input data lie and solving a supervised learning problem, but is also related to the work of Section 8.7 where we discuss simultaneously learning the generative model for input data from a code distribution prior, and solving a supervised learning problem in the code domain. As for data visualisation, it can be performed either using the model of Section 8.5 as an autoencoder, or using our approach to general-purpose kernel learning in GP-LVM developed in Section 8.6, or the results of Section 8.7.

8.4 Probabilistic Kernel PCA

The standard probabilistic interpretation of PCA introduced by Bishop (1999) posits the following generative model for inputs $x_i \in \mathbb{R}^d$:

$$x_i = m + W z_i + \epsilon_i, \quad z_i \sim \mathcal{N}(0, I), \qquad \epsilon_i \sim \mathcal{N}(0, \sigma^2 I), \quad \epsilon_i \perp z_i, \tag{8.51}$$

where W is a $d \times q$ matrix, $q < d$, $m, \epsilon_i \in \mathbb{R}^d$ and $z_i \in \mathbb{R}^q$. The marginal distribution of inputs is then

$$x_i \overset{\text{i.i.d}}{\sim} \mathcal{N}(m, \boldsymbol{C}), \quad \boldsymbol{C} = WW^T + \sigma^2 I. \tag{8.52}$$

Bishop (1999) showed that maximum likelihood estimates of m, σ^2 and W can be obtained in closed-form and read:

$$\bar{m} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \bar{\sigma}^2 = \frac{1}{d-c}\sum_{j=c+1}^{d} \lambda_j, \quad \bar{W} = \boldsymbol{U}_c(\boldsymbol{\Lambda}_c - \bar{\sigma}^2 I)^{\frac{1}{2}}, \tag{8.53}$$

where $(\lambda_j)_{1 \leq j \leq d}$ is the decreasing sequence of eigenvalues of the sample covariance matrix

$$\boldsymbol{S} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{m})(x_i - \bar{m})^T,$$

and the c column vectors in the $d \times c$ matrix $\boldsymbol{U}_c$ are the principal eigenvectors of $\boldsymbol{S}$, with eigenvalues $\lambda_1, \ldots, \lambda_c$ in the $c \times c$ diagonal matrix $\boldsymbol{\Lambda}_c$.

Remark 8.2 *It can be shown that the conditional distribution of z_i given x_i is available in closed form and reads*

$$z_i \mid x_i \overset{ind.}{\sim} \mathcal{N}\left(\boldsymbol{M}^{-1}W^T\left(x_i - m\right), \sigma^2\boldsymbol{M}^{-1}\right), \quad \boldsymbol{M} = W^TW + \sigma^2 I, \tag{8.54}$$

where we use the notation 'ind.' to denote independence. When parameters are estimated by maximum likelihood, the average term

$$\boldsymbol{M}^{-1}W^T\left(x_i - m\right)$$

is approximately equal to the coordinates of the orthogonal projection of $x_i - m$ onto the subspace with (orthogonal) basis the normalised eigenvectors (i.e. the columns of $\boldsymbol{U}_c$) scaled by the inverse of the square-root of the corresponding eigenvalues, while the covariance matrix $\sigma^2\boldsymbol{M}^{-1}$ accounts for the amount of information not captured by the c principal components. For instance, in the limit case $d = c$, $\sigma^2 = 0$, $\boldsymbol{M} = \boldsymbol{\Lambda}_d$, the code is fully deterministic conditional on x_i and reads

$$z_i = \boldsymbol{U}_d^T\boldsymbol{\Lambda}_d^{-\frac{1}{2}}\left(x_i - m\right).$$

The normalisation operated by $\boldsymbol{\Lambda}_d^{-\frac{1}{2}}$ here is an artefact of the symmetry of the i.i.d. standard normal prior on code coordinates. Without this normalisation, we would expect the resulting code coordinates, namely $\left(\boldsymbol{U}_d^T[j]\left(x_i - m\right)\right)_j$, to be a roughly decreasing sequence by virtue of $\boldsymbol{U}_d[j]$ being the j-th principal component, which wouldn't be consistent with an i.i.d. prior.

The probabilistic interpretation of PCA can be thought of as learning a structured multivariate Gaussian generative model for inputs, in the form of Equation (8.52). However, in most real-life cases, it would be unrealistic to assume that the data generating distribution is Gaussian. One way of departing from this hypothesis is to assume that there exists a transformation $\boldsymbol{\Xi}$, which we aim to learn, that maps an input x_i to a possibly larger but finite-dimensional feature vector $\boldsymbol{\Xi}(x_i)$ that is Gaussian. We may then perform probabilistic PCA in the feature space, applying the standard results that we recalled above directly to feature vectors $\boldsymbol{\Xi}(x_i)$. It turns out that, in this case, the maximum likelihood solution coincides with kernel PCA with reproducing kernel

$$k(x, y) = \boldsymbol{\Xi}(x)^T\boldsymbol{\Xi}(y).$$

8.4.1 Model Specification

Unlike the function learning approach of Section 8.2 and linear PCA, our probabilistic kernel PCA approach is not generative in that it does not induce a unique marginal distribution over inputs $x_1, \ldots, x_n$. In this Section, we propose a generative model for encoding or non-linear dimensionality reduction, which will serve as building block in Section 8.5, where we discuss joint learning of the support of the generating distribution of input data x_i as a low-dimensional manifold, and latent functions defined on the aforementioned low-dimensional manifold.

Specifically, we place on the kernel, namely k_e, the prior introduced in Equations (8.21 - 8.31) and depicted in Figure 8.1, which induces a prior on the resulting feature map $\boldsymbol{\Xi}_{k_e}$. We define the conditional distribution of the code z_i associated to input x_i given $x_1, \ldots, x_n$ and $\boldsymbol{\Xi}_{k_e}$ to be

$$z_i \mid \boldsymbol{\Xi}_{k_e}, x_1, \ldots, x_n \overset{\text{ind.}}{\sim} \mathcal{N}\left(\bar{M}_{k_e}^{-1} \bar{W}_{k_e}^T \left(\boldsymbol{\Xi}_{k_e}(x_i) - \bar{m}_{k_e}\right), \bar{\sigma}_{k_e}^2 \bar{M}_{k_e}^{-1}\right), \tag{8.55}$$

where $\bar{W}_{k_e}$, $\bar{m}_{k_e}$, and $\bar{\sigma}_{k_e}^2$ are maximum likelihood solutions of probabilistic PCA on input features $\boldsymbol{\Xi}_{k_e}(x_1), \ldots, \boldsymbol{\Xi}_{k_e}(x_n)$, and $\bar{M}_{k_e} = \bar{W}_{k_e}^T \bar{W}_{k_e} + \bar{\sigma}_{k_e}^2 I$. This induces a generative model

$$\boldsymbol{\Xi}_{k_e}, z_1, \ldots, z_n \mid x_1, \ldots, x_n, \tag{8.56}$$

wich we may use for model averaging, or as a building block for joint learning of a data manifold and functions defined on such a manifold.

Figure 8.4 illustrates the graphical model for this approach.

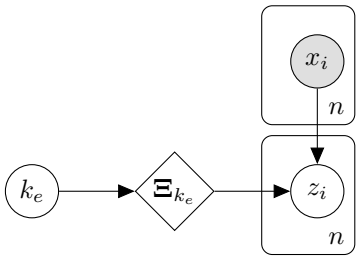

Fig. 8.4 Graphical model for probabilistic kernel PCA with general-purpose kernels.

8.4.2 Inference

Drawing i.i.d. samples from $\boldsymbol{\Xi}_{k_e}, z_1, \ldots, z_n \mid x_1, \ldots, x_n$ is straightforward in our generative model — one simply needs to draw a sample feature map $\boldsymbol{\Xi}_{k_e}$ from the prior, conditional on which codes are obtained using Equation (8.55). However,

when this generative model of codes is used in more complex modelling pipelines, for instance in Section 8.5, we may no longer be able to draw i.i.d. samples from $\boldsymbol{\Xi}_{k_e}, z_1, \ldots, z_n \mid x_1, \ldots, x_n, \Omega$, where Ω denotes additional coupling variables. For this reason, we briefly discuss a Gibbs sampling alternative below.

RJ-MCMC Updates

Sampling from

$$z_1, \ldots, z_n \mid \boldsymbol{\Xi}_{k_e}, x_1, \ldots, x_n$$

can be performed efficiently using the result of Equation (8.55). As for updating $\boldsymbol{\Xi}_{k_e}$ given $z_1, \ldots, z_n$, we note that this problem is identical to updating the kernel in collapsed GP regression (with $\mathcal{D} = (z_1, \ldots, z_n)$), for which we already proposed a solution.

Computational Cost

The resulting time complexity and memory requirement are $\mathcal{O}(ndK_{k_e} + nK_{k_e}^2 + K_{k_e}^3)$ and $\mathcal{O}(nd + nK_{k_e} + K_{k_e}^2)$ respectively, where the quadratic dependency in the number of spectral components K_{k_e} results from the need to compute and store the sample covariance matrix in the feature space, namely $\boldsymbol{S}_{k_e}$, and the term $K_{k_e}^3$ results from the SVD of $\boldsymbol{S}_{k_e}$.

8.5 Joint Encoding-Supervised Learning

The function learning approach of Section 8.2 aims at directly learning a $\mathbb{R}^Q$-valued function f of a raw input $x \in \mathbb{R}^d$. When d is very large, this approach can be computationally inefficient. For instance the dimension of the vector of RKHS coefficients $\boldsymbol{\beta}_K$ to be learned can be considerably high when d is high, even when the data live on a low-dimensional manifold. For this reason, it might be beneficial to consider learning a more structured functional representation instead, namely

$$f(x) := g(h(x)), \tag{8.57}$$

where

$$h : \mathbb{R}^d \to \mathbb{R}^c, \qquad c < d \qquad \text{and} \qquad f : \mathbb{R}^c \to \mathbb{R}^Q.$$

In other words, we posit that the data lie on a lower-dimensional manifold encoded by the transformation h, on which we learn a latent function g. Clearly, if we assume

that g and h each belong to a finite-dimensional RKHS, then it is easy to see that f also belongs to a RKHS, albeit not a finite-dimensional RKHS. One might therefore rightly argue that the representation of Equation (8.57) alone is unlikely to yield any practical gain. Indeed, the aim here is *neither* to depart from kernel methods, *nor* to consider better hypothesis spaces of functions. That being said, if h is learned so as to provide an appropriate low-dimensional manifold on which the input data lie, then the representation of Equation (8.57) can be advantageous in that, conditional on h, the learning of g will be computationally cheaper. Said differently, the key is to have in the learning of h, a sense in which, in light of the empirical distribution of training inputs x_i, $h(x)$ captures nearly all important features of x, despite being lower-dimensional.

Remark 8.3 *It is worth noting at this point that the special case $Q = d$ has the architecture of an autoencoder, and will effectively be one when, additionally, the likelihood function penalises the reconstruction error $||x - f(x)||$. In this case, h plays the role of an encoder, and g plays the role of a decoder. We stress that, unlike Bayesian nonparametric PCA, computing the reconstruction error does not require solving an optimisation problem; the encoder h and the decoder g are learned simultaneously.*

8.5.1 Model Specification

The generative model we use for joint encoding-supervised learning is a combination of the probabilistic kernel PCA and explicit supervised learning approaches developed in previous sections. We introduce an 'encoding kernel' k_e of the form of Equation (8.8), and denote $\boldsymbol{\Xi}_{k_e}(x)$ its feature map. For a given encoding kernel, we perform probabilistic kernel PCA on training inputs $(x_1, \ldots, x_n)$, and we define the c-dimensional codes $h(x_i) := z_i$ through the standard kernel PCA conditional distribution of Equation (8.55), which we recall below

$$z_i \mid \boldsymbol{\Xi}_{k_e}, x_1, \ldots, x_n \overset{\text{ind.}}{\sim} \mathcal{N}\left(\bar{M}_{k_e}^{-1} \bar{W}_{k_e}^T \left(\boldsymbol{\Xi}_{k_e}(x_i) - \bar{m}_{k_e}\right), \bar{\sigma}_{k_e}^2 \bar{M}_{k_e}^{-1}\right). \tag{8.58}$$

Next, we introduce a 'manifold kernel' k_m defined on $\mathbb{R}^c \times \mathbb{R}^c$, which we also take to be of the form of Equation (8.8). We posit that g belongs to the RKHS induced by k_m. We place on k_e and k_m independent priors which are both of the form previously introduced, namely Equations (8.21 - 8.31). Finally, we place on the coefficients of g in the RKHS induced by k_m, namely $\boldsymbol{\beta}_{K_m}$, a multivariate standard normal prior, and we restrict ourselves to i.i.d likelihoods. Figure 8.5 illustrates the resulting graphical model.

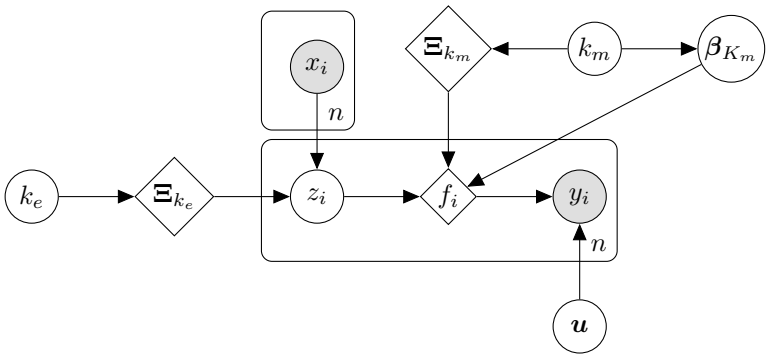

Fig. 8.5 Graphical model for joint encoding-supervised learning with general-purpose kernels.

8.5.2 Inference

Once more, we proceed in our usual Gibbs sampling fashion.

RJ-MCMC Updates

It can be noted from Figure 8.5 that updating the parameters $K_{k_m}, \boldsymbol{\theta}_{K_{k_m}}, s_{k_m}$ of the manifold kernel k_m as well as RKHS coordinates $\boldsymbol{\beta}_{K_m}$ and parameter $\boldsymbol{u}$ conditional on $z_1, \ldots, z_n$ is identical to the corresponding updates in explicit supervised learning when $x_1, \ldots, x_n$ are replaced with $z_1, \ldots, z_n$. Consequently, the sampler introduced in Section 8.2 may be reused. Similarly, conditional on $z_1, \ldots, z_n$, updating the parameters $K_{k_e}, \boldsymbol{\theta}_{K_{k_e}}, s_{k_e}$ of the encoding kernel k_e is the same problem as the corresponding probabilistic kernel PCA Gibbs sampling update we dealt with in Section 8.4. Finally, the pdf of $z_1, \ldots, z_n$ given everything else is proportional to

$$\prod_{i=1}^{n} \mathcal{N}\left(z_i \middle| \bar{M}_{k_e}^{-1} \bar{W}_{k_e}^T \left(\boldsymbol{\Xi}_{k_e}(x_i) - \bar{m}_{k_e}\right), \bar{\sigma}_{k_e}^2 \bar{M}_{k_e}^{-1}\right) p\left(y_i \;\middle|\; \boldsymbol{\Xi}_{k_m}(z_i)^T \boldsymbol{\beta}_{K_{k_m}}, \boldsymbol{u}\right). \tag{8.59}$$

Hence, each z_i can be updated in *parallel* using Elliptical Slice Sampling.

Remark 8.4 *This last update is perhaps the easiest way to see the benefits of the joint encoding-supervised learning approach compared to direct supervised learning. Indeed, we note that the learned codes z_i don't just aim at improving the model fit through the likelihood term $p\left(y_i \;\middle|\; \boldsymbol{\Xi}_{k_m}(z_i)^T \boldsymbol{\beta}_{K_{k_m}}, \boldsymbol{u}\right)$ under the budget constraint $c \ll 4K_{k_e}$, the term $\prod_{i=1}^{n} \mathcal{N}\left(z_i \middle| \bar{M}_{k_e}^{-1} \bar{W}_{k_e}^T \left(\boldsymbol{\Xi}_{k_e}(x_i) - \bar{m}_{k_e}\right), \bar{\sigma}_{k_e}^2 \bar{M}_{k_e}^{-1}\right)$ also aims at capturing in z_i the essence of the input data generating distribution as it penalises deviations from the probabilistic kernel PCA solution.*

Prediction

For a test input x, we construct consistent estimates of predictive mean $\mathrm{E}\left(f(x)|\mathcal{D}\right)$ and predictive variance $\mathrm{Var}\left(f(x)|\mathcal{D}\right)$, in the same spirit as direct and explicit supervised learning. First, we collect samples $\left(\boldsymbol{\Xi}_{k_e}, \bar{W}_{k_e}, \bar{m}_{k_e}, \bar{\sigma}^2_{k_e}, \bar{M}^{-1}_{k_e}, \boldsymbol{\Xi}_{k_m}, \boldsymbol{\beta}_{K_m}\right)$ post-burn-in. Then, for every collected sample, we generate a code $h(x)$ as

$$h(x) \mid \boldsymbol{\Xi}_{k_e}, x_1, \ldots, x_n \;\sim\; \mathcal{N}\left(\bar{M}^{-1}_{k_e}\bar{W}^T_{k_e}\left(\boldsymbol{\Xi}_{k_e}(x) - \bar{m}_{k_e}\right), \bar{\sigma}^2_{k_e}\bar{M}^{-1}_{k_e}\right), \tag{8.60}$$

which corresponds to the draw

$$\tilde{f}(x) := \boldsymbol{\Xi}_{k_m}(h(x))^T\boldsymbol{\beta}_{K_m}. \tag{8.61}$$

Finally, we compute sample mean and variance of $\tilde{f}(x)$, which constitute consistent estimates of predictive mean and variance. The random code $h(x)$ may also be used for data visualisation purposes, where the randomness in $h(x)$, which we can illustrate as a video, reflects both the amount of information about the data generating distribution not captured by the c principal components and the posterior uncertainty about the choice of kernel.

Computational Cost

At training time, updating $z_1, \ldots, z_n$ does not constitute the bottleneck of inference. Consequently, asymptotic resource requirements are the sum of the requirements for direct and explicit supervised learning and the requirements for probabilistic kernel PCA, as previously discussed. At prediction time, memory requirement and time complexity are both independent from the size of the training data set, and grow linearly in c, K_{k_e}, K_{k_m}, and in the number of post-burn-in samples used.

8.6 Gaussian Process Latent Variable Model

The Gaussian process latent variable model (GP-LVM) was introduced by Lawrence (2003) as a non-linear extension of dual probabilistic PCA for non-linear dimensionality reduction. Dual probabilistic PCA posits the same generative model for inputs as probabilistic PCA

$$x_i = m + Wz_i + \epsilon_i, \qquad \epsilon_i \;\sim\; \mathcal{N}\left(0, \sigma^2\right)$$

but rather than placing a prior on z_i and infering W like probabilistic PCA does, dual probabilistic PCA places a prior on W, namely that the rows of W are i.i.d. standard normal

$$W[j] \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I), \tag{8.62}$$

and infers z_i. After marginalising out W, we get

$$x_i[j] - m[j] \overset{\text{i.i.d.}}{\sim} \mathcal{N}\left(0, z_i^T z_i + \sigma^2\right). \tag{8.63}$$

More generally, $(x_i)_i$ is a vector-valued Gaussian process with independent coordinates, the j-th of which is indexed on the space of codes, with constant constant mean function $m[j]$, and covariance function $(x, y) \to x^T y + \sigma^2$. The idea of GP-LVM is to relax the linearity requirement in the relation between x_i and z_i, and consider instead the generative model

$$x_i \sim m + f(z_i) + \epsilon_i, \qquad f[j] \overset{\text{i.i.d.}}{\sim} \mathcal{GP}(0, k_g), \tag{8.64}$$

where the kernel k_m is to be learned. The resulting marginal likelihood reads

$$p\left(\boldsymbol{x}|\boldsymbol{z}, m, k_g, \sigma^2\right) = \prod_{j=1}^{d} \mathcal{N}\left(\boldsymbol{x}[j] \mid m[j]\mathbb{1}_n, \boldsymbol{K} + \sigma^2 I\right) \tag{8.65}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)$, $\boldsymbol{z} = (z_1, \ldots, z_n)$, $\mathbb{1}_n$ is the n-dimensional vector with coordinates equal to 1, and $\boldsymbol{K} = [k_g(z_i, z_j)]_{1 \leq i,j \leq n}$ is the Gram matrix. In the original paper, Lawrence (2003) proposed learning codes z_i, mean vector m, noise variance σ^2 and kernel hyper-paramters simultaneously by maximising the foregoing marginal likelihood using gradient based techniques.

Remark 8.5 *Like probabilistic PCA, GP-LVM implies a Gaussian generative model for the data. However, while probabilistic PCA posits that* samples *are i.i.d., GP-LVM posits that samples are distributed differently, but exhibit correlations that reflect proximities of their representations in the latent space.*

8.6.1 Model Specification

For a full Bayesian treatment, we place priors on codes z_i and kernel k_g. We take kernel k_g to be of the form of Equation (8.8). We place i.i.d. standard Gaussian priors on z_i, and our familiar general-purpose kernel prior on k_g (see Equations (8.21-8.31)).

We note that the prior GP-LVM places over f is equivalent to writing

$$f(z) = \boldsymbol{\beta}_{K_g} \boldsymbol{\Xi}_{k_g}(z), \qquad \boldsymbol{\beta}_{K_g}[j] \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, I), \qquad k_g(x, y) = \boldsymbol{\Xi}_{k_g}(x)^T \boldsymbol{\Xi}_{k_g}(y) \tag{8.66}$$

where $\boldsymbol{\beta}_{K_g}[j]$ denotes the j-th row of the matrix $\boldsymbol{\beta}_{K_g}$. Figure 8.6 illustrates the resulting graphical model.

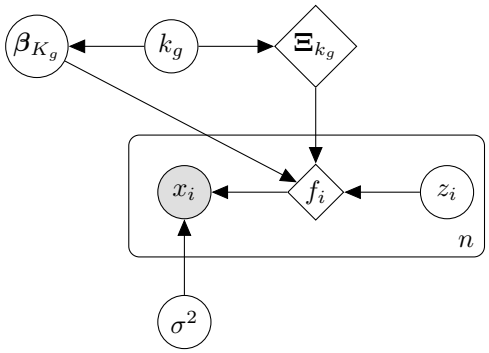

Fig. 8.6 Graphical model for Gaussian process latent variable modelling with general-purpose kernels.

8.6.2 Inference

Our aim is to sample from the posterior distribution

$$k_g, \boldsymbol{\beta}_{K_g}, \sigma^2, z_1, \ldots, z_n \mid x_1, \ldots, x_n. \tag{8.67}$$

As usual, we proceed in a Gibbs sampling style.

RJ-MCMC Updates

Conditional on $z_1, \ldots, z_n$, updating remaining variables is identical to the explicit supervised learning problem previously discussed. More precisely it is the multi-output GP regression problem with labels $\mathcal{D} = (x_1, \ldots, x_n)$, inputs $(z_1, \ldots, z_n)$, and d i.i.d. Gaussian white noises with variance σ^2. We may therefore reuse the updates of Section 8.2. The distribution of $z_1, \ldots, z_n$ conditional on remaining parameters has pdf proportional to

$$\prod_{i=1}^{n} \mathcal{N}(z_i|0, I)\, \mathcal{N}\left(x_i|\boldsymbol{\beta}_{K_g} \boldsymbol{\Xi}_{k_g}(z_i), \sigma^2 I\right), \tag{8.68}$$

which is a form suitable for an Elliptical Slice Sampling update.

Computational Cost

Updating $z_1, \ldots, z_n$ presents no additional computational challenge. Consequently, resource requirements are the same as in explicit supervised learning, which we have already discussed.

8.7 Joint Generative-Supervised Learning

In our last application, we are interested in learning from training data $(x_1, y_1, \ldots, x_n, y_n)$, how to sample both an unseen input x and its corresponding label y from the data generating distribution.

We have already introduced all ingredients needed to construct the generative model for this task. In Section 8.6 we argued that a typical input data generating distribution is likely supported on a manifold whose dimension is lower than that of the corresponding input space, and we discussed how to learn a parametrisation of this lower dimensional manifold using GP-LVM with general-purpose kernels. To sample from the data generating distribution, we simply need to sample latent variable z from a standard normal prior, and sample x from the conditional given by Equation (8.65). As for the generative model for labels y_i, we use the generative model for supervised learning introduced in Section 8.2, but with latent variables as inputs. The resulting graphical model is depicted in Figure 8.6.

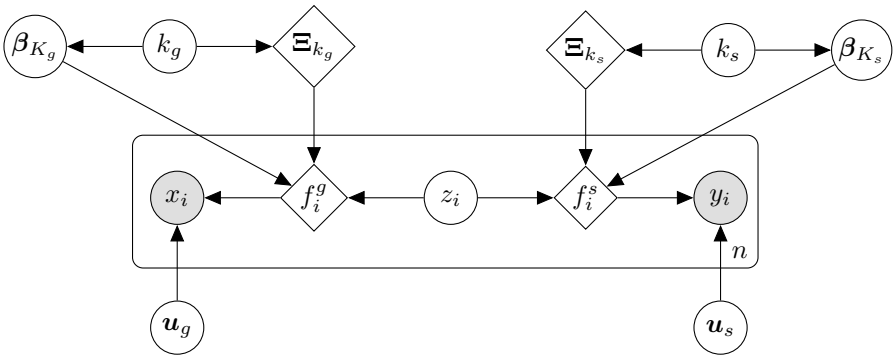

Fig. 8.7 Graphical model for joint generative-supervised learning with general-purpose kernels. $\boldsymbol{u}_g$ represents the noise variance σ^2 in GP-LVM.

8.7.1 Inference

Our aim is to sample from the posterior distribution

$$k_g, k_s, \boldsymbol{u}_g, \boldsymbol{u}_s, \boldsymbol{\beta}_{K_g}, \boldsymbol{\beta}_{K_s}, z_1, \ldots, z_n \mid x_1, y_1, \ldots, x_n, y_n. \tag{8.69}$$

As usual, we proceed in a Gibbs sampling style.

RJ-MCMC Updates

Conditional on $z_1, \ldots, z_n$, updating the generative block[8] and the supervised block[9] can be done independently and in parallel. The update of the generative block was already discussed in Section 8.6, and the update of the supervised block is identical to the RJ-MCMC Gibbs cycle introduced in Section 8.2 (replacing x_i with z_i). Finally, the distribution of $z_1, \ldots, z_n$ given all remaining variables has pdf proportional to

$$p\left(y_1, \ldots, y_n \mid \boldsymbol{\beta}_{K_s} \boldsymbol{\Xi}_{k_s}(z_1), \ldots, \boldsymbol{\beta}_{K_s} \boldsymbol{\Xi}_{k_s}(z_n)\right) \prod_{i=1}^{n} \mathcal{N}\left(z_i | 0, I\right) \mathcal{N}\left(x_i | \boldsymbol{\beta}_{K_g} \boldsymbol{\Xi}_{k_g}(z_i), \sigma^2 I\right), \tag{8.70}$$

which is of a form suitable for an Elliptical Slice Sampling update. When additionally the supervised likelihood $p\left(y_1, \ldots, y_n \mid \boldsymbol{\beta}_{K_s} \boldsymbol{\Xi}_{k_s}(z_i)\right)$ is i.i.d., as it is commonly the case, in regression and classification problems for instance, we may update all z_i independently and in parallel.

Computational Cost

Once more, updating $z_1, \ldots, z_n$ presents no additional computational challenge. Consequently, resource requirements are the same as in explicit supervised learning, which we have already discussed.

8.8 Flashback: Semi-Supervised Learning

In many real-life situations, labelling data is expensive. The operator will typically have access to n inputs $x_1, \ldots, x_n$, for instance images, only $l < n$ of which, say $x_1, \ldots, x_l$, will have labels, say $y_1, \ldots, y_l$. In such a situation, it would be inefficient to throw away unlabelled inputs $x_{l+1}, \ldots x_n$, and perform supervised learning on labelled ones. Indeed, unlabelled inputs may contribute towards improving the learning of the manifold on

[8] The block of variables on the left hand side of z_i in Figure 8.6.
[9] The block of variables on the right hand side of z_i in Figure 8.6.

which input data lie, which might simplify the supervised learning problem in the latent space, thereby lowering the number of labels required to generalise well. Paradoxically, having many unlabelled inputs compensates to some extent not having enough labelled ones. This is the domain of semi-supervised learning.

The approaches discussed in Sections 8.5 and 8.7 for jointly performing encoding and supervised learning, and jointly performing generative and supervised learning, can be easily adapted to semi-supervised learning. All we need to do is remove the generative block from the corresponding graphical models (Figures 8.5 and 8.7) for codes of unlabelled inputs, namely $z_{l+1}, \ldots, z_n$. Inference schemes are unchanged, except for updates of latent variables z_i. In the case of joint encoding-supervised learning (Section 8.5), we simply need to remove $z_{l+1}, \ldots, z_n$ from the inference pipeline at training time. We stress however that $z_1, \ldots, z_l$ will still depend on all input points, including $x_{l+1}, \ldots, x_n$. In the case of joint generative-supervised learning (Section 8.7), the conditional distribution of $(z_1, \ldots, z_n)$ given everything else becomes

$$p\left(y_1, \ldots, y_l \mid \boldsymbol{\beta}_{K_s} \boldsymbol{\Xi}_{k_s}(z_1), \ldots, \boldsymbol{\beta}_{K_s} \boldsymbol{\Xi}_{k_s}(z_l)\right) \prod_{i=1}^{n} \mathcal{N}\left(z_i|0, I\right) \mathcal{N}\left(x_i|\boldsymbol{\beta}_{K_g} \boldsymbol{\Xi}_{k_g}(z_i), \sigma^2 I\right),$$

which, once more, is suitable for an Elliptical Slice Sampling update.

8.9 Experiments

In the interest of brevity, in this section we focus on illustrating the main two points developed throughout this chapter, namely that i) one is usually better-off learning model complexity from the data than setting it beforehand, and the approach we propose is useful in that respect, and ii) learning nonstationarity from the data using our approach does indeed add value compared to stationary alternatives. Then we briefly illustrate a data visualisation application.

8.9.1 Supervised Learning

To illustrate the foregoing main two points, we select 4 small to medium size datasets in the UCI Machine Learning repository, namely Bike Sharing (regression), Diabetic (classification), EEG (classification), and Pima (classification). To illustrate the effects of nonstationarity and model complexity learning, we compare the direct supervised learning inference scheme introduced in Section 8.2 —which aims at automatically learning model complexity and nonstationarity from the data—, with the BaNK model

of Oliva et al. (2016) —which is restricted to stationary kernels with a fixed model complexity—, and with the inference scheme of Section 8.2, but where we fix $s = 0$, thereby allowing for model complexity learning, while enforcing stationarity. For every pair dataset-model, we perform 10 random experiments, selecting 9/10-th of the dataset uniformly at random for training, and using the rest for testing, and we report metrics as mean $\pm$ one standard error. For BaNK, we use $K = 50$ random features or spectral components, and for the other two methods, we choose the parameters of the Poisson-Gamma prior on K such that it has mean 50 and variance 250 (which we hope is large enough to emulate uninformativeness), and we initialize K to 50. In each experiment, we run 10,000 Gibbs iterations, we discard the first 5,000 as 'burn-in', and we apply a 1-in-100 thinning rate to the remaining 5,000 so as to mitigate serial-correlation of samples. Results are reported in Table 8.1 and illustrated in Figure 8.8.

First of all, we note from Table 8.1 that GSK outperforms competing alternatives in all 4 cases. Crucially, our approach is able to learn when a model complexity higher than the baseline $K = 50$ is warranted by the data (e.g. Bike Sharing and EEG), in which case the resulting increase in execution time is a fair price to pay for improved accuracy. Our approach is also able to learn when our baseline model complexity is excessive (e.g. Diabetic and Pima), in which case it is able to speed-up execution by reducing model complexity, while maintaining superior accuracy. The cumulative probability mass functions of posterior distributions over K are illustrated in Figure 8.8.

Second, by comparing the results of GSK and stationary GSK in Table 8.1, it can be seen that nonstationarity learning indeed does add value. Interestingly, in all 4 instances, the posterior mean of the nonstationarity switch parameter s of the GSK model is much lower than its prior mean ($s = 0.5$), even though stationary GSK, which corresponds to $s = 0$, performs worse than GSK. This suggests that thinking in terms of shrinkage of the off-diagonal mass of the spectral bimeasure might be a better perspective than the usual stationarity/nonstationarity dichotomy.

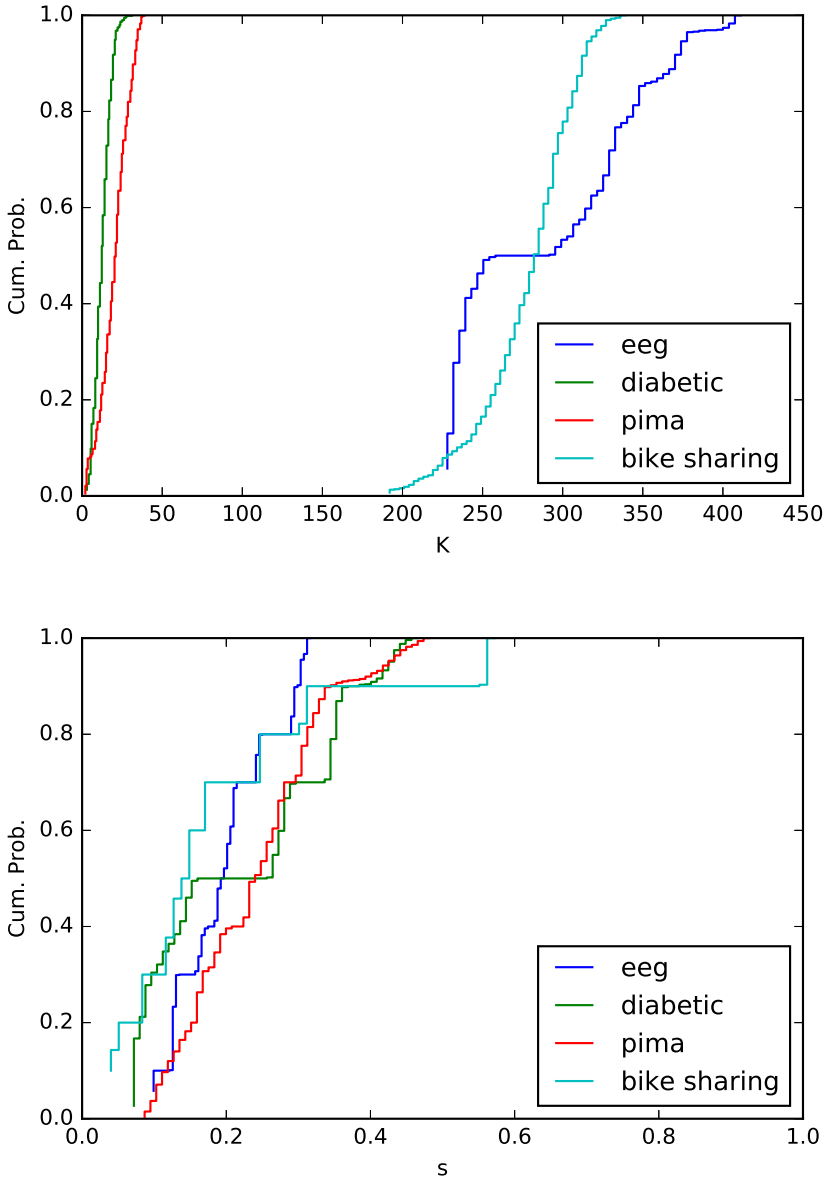

Fig. 8.8 Posterior cumulative density function (CDF) of model complexity parameter K and nonstationarity parameter s on standard regression and classification tasks from the UCI Machine Learning dataset repository, as described in Section 8.9.

Table 8.1 Comparison of 3 Bayesian nonparametric kernel learning approaches on 4 UCI regression and classification tasks. GSK is the approach we introduce in this Chapter, Stat. GSK corresponds to the special case where the nonstationarity parameter s is fixed and equal to 0.0 (i.e. stationarity GSK), and BaNK is the model of Oliva et al. (2016). Where $n = x/y$, the dataset is made of $x + y$ observations, which were split, 10 times uniformly at random, into a training dataset of size x and a test dataset of size y. Metrics are presented as (posterior) mean +/- standard error over the 10 random experiments. For the regression task (i.e Bike Sharing), 'Error' is the root mean square error, normalised by the standard deviation of the output, and for classification tasks, 'Error' is the classification error. K is the number of spectral components, T(s) is the total CPU time in seconds (across multiple CPU cores) taken by 1 Gibbs iteration, and s is the nonstationarity switch parameter.

		Error	K	T(s)	s
Bike Sharing (n=15642/1737, d=14)	GSK	$\mathbf{0.24 \pm 0.01}$	281.97 ± 18.02	45.72 ± 1.25	0.20 ± 0.02
	Stat. GSK	0.34 ± 0.01	238.58 ± 23.38	19.53 ± 0.40	0.00 ± 0.00
	BaNK	0.32 ± 0.03	50.00 ± 0.00	16.49 ± 0.77	0.00 ± 0.00
Diabetic (n= 1036/115, d= 19)	GSK	$\mathbf{0.39 \pm 0.02}$	13.36 ± 1.62	0.10 ± 0.00	0.23 ± 0.04
	Stat. GSK	0.42 ± 0.03	14.41 ± 1.72	0.10 ± 0.00	0.00 ± 0.00
	BaNK	0.40 ± 0.02	50.00 ± 0.00	2.52 ± 0.06	0.00 ± 0.00
EEG (n=13482 / 1498, d= 14)	GSK	$\mathbf{0.20 \pm 0.00}$	293.15 ± 18.02	9.01 ± 0.28	0.20 ± 0.02
	Stat. GSK	$\mathbf{0.20 \pm 0.00}$	275.28 ± 23.07	8.67 ± 0.34	0.00 ± 0.00
	BaNK	0.39 ± 0.00	50.00 ± 0.00	2.10 ± 0.03	0.00 ± 0.00
Pima (n=692 / 76 , d= 8)	GSK	$\mathbf{0.26 \pm 0.01}$	21.25 ± 2.86	0.13 ± 0.04	0.25 ± 0.03
	Stat. GSK	0.28 ± 0.01	22.30 ± 1.97	0.08 ± 0.00	0.00 ± 0.00
	BaNK	$\mathbf{0.26 \pm 0.02}$	50.00 ± 0.00	0.54 ± 0.01	0.00 ± 0.00

8.9.2 Data Visualisation

Lastly, we consider illustrating the benefits of our approach for data visualisation. To do so, we reuse the EEG dataset previously mentioned, which we recall is made of 14-dimensional inputs with binary labels. As is customary for data visualisation tasks, our goal is to represent input data in 2D, with the hope of uncovering within-label structures. For this, we adopt the joint supervised-encoding scheme of Section 8.5, with code budget $c = 2$, and we visualise the data in the learned 2D code space. As competing approaches, we consider PCA and GP-LVM with the RBF kernel, both with the same code budget.[10] The results are illustrated in Figure 8.9.

It can be seen at a glance in Figure 8.9 that our approach (top row) does a considerably better job at clustering inputs with similar labels in the latent space. This is perhaps not surprising. In effect, neither PCA nor GP-LVM uses labels. PCA simply encodes points by projecting them onto the two directions in which the data vary the most, which does not guarantee clustering of points with similar labels in the latent space. As for GP-LVM, although it provides a way in which euclidean proximity in the latent space may result in euclidean proximity in the original space,[11] it doesn't aim at ensuring that euclidean proximity in the latent space implies label similarity. It is also worth noting from Figure 8.9 that, contrary to joint encoding-supervised learning, the encoding schemes operated by PCA and GP-LVM can hardly be used as a pre-training step for classification on this dataset, as the resulting codes don't appear to be separable by a smooth curve.

[10] For GP-LVM we use the 'BayesianGPLVM' GPy implementation.
[11] When the kernel is continuous and the noise variance is small.

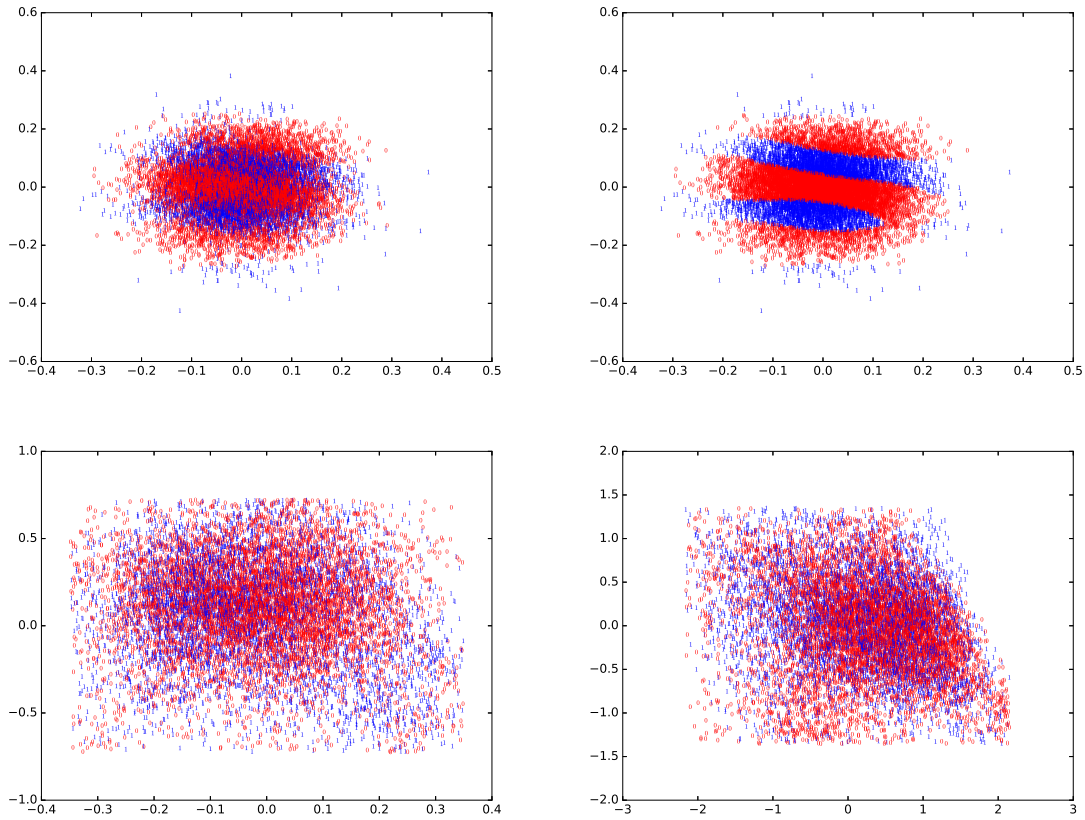

Fig. 8.9 Visualisation of the EEG dataset (from the UCI repository) using the Joint Encoding-Supervised scheme of Section 8.5 (top row), PCA (bottom left) and GP-LVM with the RBF kernel (bottom right). Each point represents a training 14-dimensional input, its coordinates represent the associated two-dimensional code z_i, and labels reflect the corresponding classes. In the case of Joint Encoding-Supervised Learning, codes z_i correspond to a random state of the Markov chain; the top left figure illustrates ground-truth labels, whereas the top right figure illustrates labels as predicted by the 'supervised component' of the joint encoding-supervised learner.

8.10 Discussion

In this Chapter we propose inference techniques for performing supervised, semi-supervised and unsupervised learning with general-purpose kernels, with an emphasis on i) model complexity learning, ii) nonstationarity learning, and iii) scalability. Nonetheless, we barely scratch the surface of what can be achieved with general-purpose kernels, and the proposed methods can be extended in many ways.

First of all, in this Chapter we heavily rely on Elliptical Slice Sampling (ESS) for updates as it is tuning-free. An alternative would be to use the Metropolis-Hastings (MH) algorithm with proposal constructed using Variational Bayes, or VB. This approach would be less sensitive to fine-tuning than the MH algorithm with proposals chosen before run-time and that do not dependent on the data. MH with VB proposals is also advantageous in that, when the likelihood is i.i.d., we may use Stochastic Variational Bayes (Hoffman et al. (2013)). In this case, the full likelihood only needs to be evaluated once per update, to compute the acceptance ratio, and only a small subset of the data is used to construct the proposal. It's worth noting that it is not necessary that the VB approximation converges, as any intermediary approximate distribution would still be a valid proposal distribution. A possible heuristic would be to stop the VB approximation when the lower bound does not improve by a certain percentage.

Another alternative is to rewrite all joints with Gaussian priors as

$$p(\mathcal{D}, \mathbf{x}) = p(\mathcal{D}|\mathbf{x})\frac{p(\mathbf{x})}{\hat{p}(\mathbf{x})}\hat{p}(\mathbf{x}) := \tilde{p}(\mathcal{D}|\mathbf{x})\hat{p}(\mathbf{x}) \tag{8.71}$$

where $\mathbf{x}$ is a generic latent variable with Gaussian prior $p(\mathbf{x})$, and $\hat{p}(\mathbf{x})$ is a Gaussian VB approximation to the posterior $p(\mathbf{x}|\mathcal{D})$, and to perform Elliptical Slice Sampling using the right-most expression. This could improve on straight-ESS when prior and posterior modes are too apart.

In most cases, for a given model complexity (i.e. the number of spectral components K), one could also perform variational inference directly on the whole model (conditional on K). However, the approaches presented in this Chapter, coupled with VB, either through MH proposals or ESS updates, would be preferable to direct VB, in that they seamlessly allow for model complexity learning, which is a major advantage of RJ-MCMC methods over competing approaches, including VB.

Finally, although we adopted a Bayesian nonparametric approach throughout this Chapter, it is worth recalling that finite-dimensional RKHSs are effectively parametric families of functions, and most of the ideas discussed herein can be revisited with a

frequentist and parametric perspective. That said, such an approach is unlikely to handle model complexity learning, and might be prone to overfitting.

In future works, we will provide empirical evidence of the efficacy of the proposed algorithms and suggested extensions on open AI challenges.

Chapter 9

From Kernel Learning to Deep Learning

> "We can complain because rose bushes have thorns, or rejoice because thorn bushes have roses."
>
> *Abraham Lincoln*

Thus far we have been focusing on addressing the two limitations of kernel methods, namely their lack of scalability and their lack of flexibility. In this chapter we go a step further towards bridging the gap between kernel methods and deep learning, and we provide a sense in which deep learning can be regarded as a special type of kernel learning. We conclude that kernel learning researchers should see in recent deep learning breakthroughs an indication that future AI challenges may be solved, in a more scalable, robust, and principled manner, using kernel methods.

In previous chapters we've analysed the limitations of kernel methods in handling large scale and complex inference problems, applications in which deep learning has recently generated numerous breakthroughs, and we have proposed theoretical and algorithmic solutions. Nonetheless, our development so far has been completely unrelated to deep learning itself. At the end of the day, in most machine learning tasks, both kernel methods and deep learning simply aim at learning latent functions from training data. Moreover, the measures of the extent to which a given candidate latent function is appropriate for the machine learning task of interest, that are used by both methods, are widely interchangeable, and primarily differ by their names

(i.e. *negative log-likelihood* for Bayesian kernel methods or *loss function* for frequentist kernel methods and deep learning). Consequently, from a high level perspective, the only two differences between deep learning and kernel methods are the hypothesis space of latent functions postulated by each method, and the function learning algorithm used by each method to explore its hypothesis space of candidate functions, using evidence provided by training data.

The difference between the hypothesis spaces of candidate functions of both methods has been used by deep learning pioneers for a long time to justify the success of deep learning (Bengio et al. (2013); LeCun et al. (2015)). They claim that kernel methods are 'shallow' because they are thought to yield latent functions that are not complex enough to capture patterns as intricate as the ones deep learning can capture. For instance Bengio et al. (2013) says of kernel methods that '[...] most of these algorithms only exploit the principle of *local generalization*, i.e., the assumption that the target function (to be learned) is smooth enough, so they rely on examples to explicitly map out the wrinkles of the target function [...]'. But are there *really* tasks for which the hypothesis spaces of candidate functions expressed by deep neural networks are inherently more suitable/flexible than can be achieved with kernel methods?

In this chapter we address this aforementioned question. We prove in particular that the hypothesis space of candidate functions expressed by a neural network, no matter how deep, can always be completed into a 'pseudo-RKHS', a mathematical concept we define in the following section, which for most practical purposes is the same as an RKHS. This means that the hypothesis space expressed by a neural network, no matter how deep, can always be expressed using a kernel method. Moreover, we prove that the reproducing kernel of the completed pseudo-RKHS is continuous and bounded, which implies that it can be learned using a general-purpose family of kernels.

9.1 Related Work

Several attempts have been made to combine deep learning and kernel learning, but often they end up aggregating the limitations of both approaches.

One such attempt is the Deep Gaussian Process model introduced by Damianou and Lawrence (2013). The authors proposed constructing a functional prior by stacking up identical elementary Gaussian processes hierarchically so as to mimic deep neural networks. The hope is that the hierarchical nature of this construction will facilitate the learning of complex features. This model raises some fundamental questions. When all building block GPs are mean-zero, the resulting deep GP will provably be a mean-zero

stochastic process, with some covariance function. Moreover, we already argued that the space of all continuous bounded covariance functions contains all covariance functions of practical interest. Therefore, if deep GPs result in more accurate inference than GPs with general-purpose kernels, then it would have to be because of the nature of their (non-Gaussian) finite-dimensional marginals. However, not only are the marginals of deep GPs tedious to characterize analytically, but it is unclear in what sense should one expect them to be more flexible/suitable than multivariate Gaussians. Furthermore, we are not aware of principles by which the architecture of the network of GPs could be chosen, in the same way that for the most part choosing the number of layers and nodes in a neural network is in practice driven by heuristics and the experience of the implementor.

Another approach consists of engineering features using a deep neural network whose output layer is then used as input to a kernel method. For instance, Wilson et al. (2016) used the output layer of a deep neural network as input to a spectral mixture kernel, and argued that the neural network allows for 'nonstationarity learning'. Considering that both deep learning and kernel methods aim at learning features/representations, it seems odd that one would combine the two without determining what property would the combination have that can't be obtained using a single method. In this regards, 'nonstationarity' is certainly not the answer, as general-purpose kernels allow for nonstationarity learning as flexibly as required. More importantly, this approach is subject to the limitations of deep learning and kernel methods, namely the lack of a principled approach for determining the architecture of the neural network, and issues of flexibility and scalability on the kernel component.

In the rest of this Chapter, we compare deep learning and kernel learning. In particular, we discuss why kernel methods can perform at least as well as deep learning, and we argue that kernel methods can be a better alternative to deep learning.

9.2 Deep Learning Meets Kernel Learning

We begin by introducing a notion similar to reproducing kernel Hilbert spaces (RKHS), which we denote pseudo-RKHS.

Definition 9.1 *(Pseudo-RHKS) Let $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ be a Mercer kernel. We denote pseudo-RKHS with reproducing kernel k, namely $\hat{\mathcal{H}}_k$, the completion of the pre-RKHS*

$$\mathcal{H}_{0k} = span\left(\{k(.,x),\, x \in \mathcal{X}\}\right),$$

with respect to the pointwise convergence of functions.

Remarks: It is well known that the RKHS with reproducing kernel k, namely $\mathcal{H}_k$, can be obtained as the completion of the pre-RKHS $\mathcal{H}_{0k}$ with respect to the RKHS norm, so that the main difference between RKHS and pseudo-RHKS is the topology with respect to which each set is complete. Moreover, recalling that convergence in RKHS norm implies pointwise convergence, it follows that an RKHS is contained in the corresponding pseudo-RKHS, more precisely we have

$$\mathcal{H}_{0k} \subseteq \mathcal{H}_k \subseteq \hat{\mathcal{H}}_k. \tag{9.1}$$

Furthermore, when the RKHS $\mathcal{H}_k$ is finite-dimensional, pointwise convergence and convergence in RKHS norm are the same, and $\mathcal{H}_{0k} = \mathcal{H}_k = \hat{\mathcal{H}}_k$. As it is not uncommon to consider finite-dimensional feature space approximations (e.g. random Fourier features), for most practical purposes, there isn't much difference between these three spaces of functions.[1]

Going back to our comparison between kernel methods and deep learning, in order to show that kernel methods have the potential of being as flexible as deep learning, we prove that any neural network, no matter how deep, can always be completed into a pseudo-RKHS whose reproducing kernel is continuous and bounded.

Theorem 9.2 *(Pseudo-RKHS completion of neural networks) Let $\mathcal{X}$ be a compact subset of a measurable space. Let*

$$\mathcal{F} = \left\{ f_\theta : \mathcal{X} \to \mathbb{R}, \quad \theta \in \mathbb{R}^N, \ N \in \mathbb{N}^* \right\}$$

be the hypothesis space of functions expressed by a neural network with weight parameters $\theta \in \mathbb{R}^N$, and Lipschitz activation function(s). There exists a pseudo reproducing kernel Hilbert space $\hat{\mathcal{H}}_k$, with continuous and bounded reproducing kernel k, such that

$$\mathcal{F} \subseteq \hat{\mathcal{H}}_k.$$

Proof See Appendix F.1. ■

Let us take a minute to analyse this result. First of all, the only conditions the theorem requires of the neural network for the pseudo-RKHS completion to exist is that its input domain be compact, and its activation function(s) be Lipschitz. The former

[1] It is worth noting however that, when the RKHS is not finite-dimensional, the RKHS can be a much smaller space than its associated pseudo-RKHS.

requirement is relatively mild given that it would be of little practical interest to require input domains to be unbounded. As for the latter, nearly all neural network activation functions used in practice are Lipschitz (e.g. ReLu, identity, arctan, tanh, Gaussian, soft sign, logistic etc) with the exception of the sign function and the binary step function, which can be replaced by the soft sign function and the logistic function (or soft step function) respectively. Moreover, the Lipschitz condition is a sufficient condition that simplifies the proof, but we conjecture that it can be relaxed. Secondly, we would like to stress that there is no condition on the architecture of the neural network; in particular the result holds irrespective of the depth of the neural network.

Remarks: Interestingly, there are infinitely many kernels k for which Theorem 9.2 holds. The alert reader that you are has probably noticed that, by definition, the pseudo-RKHS of any universal kernel (of which there are infinitely many), is the space of all continuous functions on $\mathcal{X}$, namely $\mathcal{C}^0(\mathcal{X})$.[2] The advantage of the rather lengthy constructive proof we provide in Appendix F.1 is that the constructed pseudo-RKHS is much smaller than $\mathcal{C}^0(\mathcal{X})$.

A consequence of Theorem 9.2 is that, for every one of the neural networks underpinning each of the deep learning breakthroughs we have witnessed over the past decade, there exists a continuous bounded kernel k such that, by exploring functions of the form

$$f(x) = \sum_{i=1}^{n} \beta_i k(x, x_i), \;\; x_i \in \mathcal{X}, \; \beta_i \in \mathbb{R},$$

where n could possibly be infinite, we would encounter each and every candidate function expressed by the deep neural network, including the function corresponding to the calibrated neural network. We stress however that, here, the points x_i are unrelated to training inputs, which might not form the most suitable set of support points.

9.3 Advantages of the Kernel Learning Perspective

In the previous section we provided theoretical evidence that kernel learning has the potential of performing as well as deep learning, in that deep neural network hypothesis functions may be constructed with kernels, but what practical advantages do kernel methods have over deep learning?

[2]Universality of a continous kernel means uniform density of its pre-RKHS in the space of all continuous functions, which in particular implies that its pseudo-RKHS contains $\mathcal{C}^0(\mathcal{X})$. Moreover, given that the uniform limit of continous functions is also continuous, the aforementioned pseudo-RKHS is in fact $\mathcal{C}^0(\mathcal{X})$.

The very characteristic of deep neural networks, namely the depth of the network architecture, and subsequently the number of parameters characterising its hypothesis space of candidate functions, is also one of its main limitations. Firstly, minimising the loss function over hundreds of millions of parameters is very computationally expensive; learning state-of-the-art deep neural networks might take weeks if not months on cutting-edge computing infrastructures. Secondly, optimising that many parameters requires a very large number of training samples in order to avoid overfitting and so as to generalise well to unseen test data; in other words, deep learning is *data inefficient*. These two problems limit the usefulness of deep learning applied to small and medium size problems. Thirdly, the network architecture, which determines the function space and subsequently the likelihood that it contains appropriate candidate functions, is rarely part of the learning pipeline in deep learning modules, and is typically engineered in an unprincipled way. Finally, optimization methods commonly used to calibrate deep neural networks are often very sensitive to initialisation, which is typically performed using heuristics.

That being said, as previously discussed, any of the commonly used neural networks, no matter how deep, can always be completed into a pseudo-RKHS whose reproducing kernel is continuous and bounded. As previously mentioned, there are infinitely many such pseudo-RKHSs. Of particular interest is the one that is the smallest.[3] It corresponds to adding just enough functions to the span of the original neural network to obtain pointwise completeness and the pseudo-reproducing property, and consequently, for most practical purposes, it deviates only marginally from the original neural network. Thus, if an *oracle* deep neural network yields a hypothesis space of candidate functions that contains suitable candidate functions for the task of interest, then there exists a pseudo-RKHS with continuous bounded kernel (the pseudo-RKHS completion of the oracle deep neural network) that contains equally suitable candidate functions. On one hand, learning such an oracle neural network is a tedious task as it involves learning the network architecture and the activation function(s), which is a combinatorial problem for which no suitable solution has been proposed to the best of our knowledge. On the other hand, learning a continuous bounded kernel can be achieved easily, and in a principled manner, using the results and techniques developed in Chapters 6, 7 and 8. It is also worth stressing that the results and techniques introduced in Chapters 6, 7 and 8 do not suffer from the same initialisation issue deep learning suffers from, and do not require large datasets to generalise well.

[3] We do not know it, and need not know it.

Overall, we may conclude that kernel methods have the potential of performing as well as deep learning, with the added benefit that they provide a more principled, robust, and data-efficient alternative.

Chapter 10

Conclusion

10.1 Summary

In this thesis we propose methods for tackling the two main limitations of kernel methods, namely their lack of flexibility and their lack of scalability.

In Part II we begin by focusing on Bayesian solutions to the aforementioned limitations when the data exhibit local patterns. In particular, we show that the lack of scalability of kernel methods is often the result of excessive regularity assumptions such as infinite differentiability of integrable kernels and the absence of conditional independence structure in the finite-dimensional marginals of commonly used functional priors such as Gaussian and Student-t processes. We empirically illustrate that i) restricting the degree of differentiability of the kernel and ii) constructing smooth stochastic processes with marginals that exhibit suitable conditional independence structures, yield methods that can scale up to sizes never considered by competing alternatives. Crucially, it happens that these two ideas for scaling up kernel methods also allow for greater flexibility, which unsurprisingly often results in improved accuracy. The contribution of Part II is to propose methods that implement these two aforementioned ideas, with applications ranging from time series forecasting and analysis, to learning intensity functions of point processes, to nonparametric regression and classification, to dynamic capital allocation in stock markets. In effect, we use these ideas in Chapter 2 to construct a scalable algorithm for learning the intensity function of a point process using Gaussian processes, a problem long considered to be 'doubly-intractable'. In Chapter 3 we exploit the conditional independence structure of p-Markov Gaussian processes to propose the first asynchronous time series model that allows for both online learning of model parameters and online forecasting, under a family of covariance functions we prove may perform as well as any alternative stationary covariance

function, and with time complexity and memory requirement that are independent from the sample size. As a generalisation, in Chapter 4 we construct a novel suite of stochastic processes, namely string Gaussian processes (string GPs), that are only once continuously differentiable and have finite-dimensional marginals that exhibit suitable conditional independence structures. In Chapter 5 we propose methods for making Bayesian inference on latent functions under string GP functional priors, which we empirically illustrate scale better than competing alternatives, and cope better with local patterns in datasets.

In Part III we generalise our discussion to all kernel methods, frequentist or Bayesian, whether or not the data can be regarded as locally homogeneous. This part of the thesis is motivated by the observation that, although it is well-known that the performance of a kernel method depends on the kernel used, the choice of kernel is often left to the user, who too often resorts to vanilla stationary kernels. Moreover, when new nonstationary positive definite kernels are proposed in the literature, they are often engineered for specific applications or datasets. Part III aims at addressing this issue with solutions that scale. In Chapter 6, we begin by discussing what it would take for a family of kernels to be suitable for any kernel methods on any datasets, property we term *general-purposeness*. We prove that if a family of kernels is pointwise dense in the family of all continuous bounded kernels, then it contains kernels that may perform as well as *any* oracle kernel of practical interest, in *any* of the commonly used kernel methods and on *any* dataset. In other words we prove that, in nearly all kernel methods, pointwise convergence of kernels implies convergence of the corresponding sequence of performances. In Chapter 7 we complement this finding by proposing a family of kernels, namely *generalized spectral kernels*, which we prove is indeed pointwise dense in the family of all continuous bounded kernels, and consequently may serve as basis for automated kernel learning. Given that the family of continuous bounded kernels is infinite-dimensional, any family of kernels that is pointwise dense in the family of all continuous bounded kernels ought to be infinite-dimensional as well, and care should be taken while deriving kernel learning algorithms for exploring such a hypothesis space of candidate kernels. One should typically navigate through finite-dimensional subspaces whose dimensions the kernel learning algorithm should allow to decrease or increase unboundedly at training time as warranted by the data. In Chapter 8 we propose such kernel learning algorithms for supervised, semi-supervised and unsupervised learning, that have linear time complexity and linear memory requirement.

10.2 Possible Extensions

The contributions of this thesis can be extended in many ways, and we encourage the reader to do so. We view this thesis as barely scratching the surface of what we consider possible with kernel methods. Notably, other approaches may be proposed for learning the dynamics of *string Gaussian processes*. More generally, other stochastic processes may be proposed that exhibit suitable degrees of smoothness and conditional independence, and that may cope with other (possibly non-numeric) types of input. Moreover, other families of kernels may well exist that are also *general-purpose*, and that provide a more efficient representation than *generalized spectral kernels*.

10.3 The Future of Kernel Learning

First and foremost, it is my belief that kernel methods are currently largely underrated, and excluded from the deep learning virtuous circle, as follows. Many machine learning breakthroughs with high commercial value are achieved and reported under the deep learning banner, thereby leading to vast amounts of money invested by major tech companies through grants, acquisitions and hires, which in turn creates a huge incentive for young researchers to jump on the deep learning bandwagon, perhaps without questioning enough what made the breakthroughs possible in the first place.

Deep learning skeptics argue that the recent success of deep learning is not due to methodological advances, as techniques used these days were mostly introduced decades ago, and that the recent success of deep learning is rather due to the fact that, half a century after the introduction of neural networks, computers have become $10,000$ times more powerful per dollar, and large scale labelled training datasets such as the ImageNet database are now available to machine learning researchers. Although this is a fair argument, it doesn't explain why alternative methods such as kernel methods haven't performed as well so far.

I personally think that, paradoxically, the success of deep learning can be attributed to its relatively low 'theoretical barrier to entry'. The theoretical construction of deep learning is relatively straightforward. It involves a neural network playing the role of a parametric family of candidate functions, and a differentiable loss function which is to be optimised. Questions pertaining to the configuration of neural networks (i.e. the choice of network architecture and link function) are often left to trial and error. Hence, the main problem any deep learning researcher has been trying to solve is *how to go about appropriately learning (the network weights)?* This question has been

tackled with a lot of pragmatic engineering, and ideas such as *backpropagation* (Kelley (1960); Rumelhart et al. (1986)), *unsupervised pre-training* (Bengio et al. (2006); Erhan et al. (2010); Hinton et al. (2006); Poultney et al. (2006)), and *dropout* (Srivastava et al. (2014)), which arguably aren't theoretically sophisticated, have had a critical impact on the success of deep learning (Erhan et al. (2010); Rumelhart et al. (1986)). Of course, this learning question may be interpreted as a non-convex optimization problem, and one may attempt to solve it through fundamental and more sophisticated contributions to the non-convex optimization literature. However, not only would this perspective be rather restrictive, for instance it would rule out practical ideas that use the geometry of neural networks such as *unsupervised pre-training* and *dropout*, but it would also be much harder.

In contrast, the theoretical construction of kernel methods requires a considerably heavier mathematical artillery. Consequently, research on kernel methods tends to attract good mathematicians who often care more about developing nice pieces of theory than solving concrete real-life problems. As for engineers working on kernel methods, they use them as tools, typically applying well known results to new problems, and they unduly do not feel concerned with advancing kernel methods as a field. That being said, the fundamental learning question previously mentioned, namely *how to go about appropriately learning (in this case, the latent function within the RKHS)*, is still as important. However, it is often overlooked by kernel learning researchers, and unduly amalgamated with an optimisation problem, which is typically required to be simple enough to have a unique analytical solution for a given kernel. This restriction can have severe practical implications.

As an illustration, let us consider the optimisation problem solved by regularised ERM, namely

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} l\left(f(x_i), y_i\right) \; + \; \lambda n ||f||_{\mathcal{H}}^2, \; \lambda \geq 0, \tag{10.1}$$

where $\mathcal{H}$ is an RKHS, l the loss function, $(\dots, x_i, \dots)$ training inputs with corresponding labels $(\dots, y_i, \dots)$. At the end of the day, this optimisation problem simply provides a way of exploring a given RKHS, in the light of some training data, so as to find a suitable candidate latent function for the supervised learning task at hand. Although the mathematical formulation of Equation (10.1) makes intuitive sense, it is by no means equivalent to the original learning question; in particular one might question the choice of the squared RKHS norm in the model complexity part of the objective function. In effect, for any increasing smooth function g, the term $g\left(||f||_{\mathcal{H}}\right)$ would make a valid model complexity part; which g should we choose? In fact, why should

the model complexity term be additive? Of course we need to start somewhere and make some assumptions, but too often the assumptions that are made at this stage aim at easing analytical derivations rather than providing a better solution to the problem at hand. For instance, the additive form

$$f^* = \underset{f \in \mathcal{H}}{\operatorname{argmin}} \sum_{i=1}^{n} l\left(f(x_i), y_i\right) \; + \; g\left(||f||_{\mathcal{H}}\right), \tag{10.2}$$

where g is an increasing function, is often chosen so that the *representer theorem* (Schölkopf and Smola (2001)) applies and the optimal solution takes the simple form

$$f^*(x) = \sum_{i=1}^{n} \beta_i k(x, x_i) \tag{10.3}$$

with k the reproducing kernel of $\mathcal{H}$, $\beta_i \in \mathbb{R}$, and where x_i are training inputs. Under the additive model complexity assumption, the optimisation problem (10.2) is equivalent to a more standard optimisation problem of the form

$$\boldsymbol{\beta}^* = \underset{\boldsymbol{\beta} \in \mathbb{R}^n}{\operatorname{argmin}} \; L(\boldsymbol{\beta}) \; + \; g\left(\sqrt{\boldsymbol{\beta}^T K \boldsymbol{\beta}}\right), \tag{10.4}$$

where K is the Gram matrix, which in certain cases such as kernel Ridge regression, admits a closed-form solution.

This analytical convenience however comes with restrictions that are often overlooked, but that can have major practical implications. For instance, one might wonder why the number of basis functions that make the optimal function as per the representer theorem (Equation (10.3)), should be equal to the number of training data points. Arguably, the number of basis functions should depend on how fast the kernel k varies over its domain. For example, if $k(x, y) = e^{-10^{-10}||x-y||^2} \approx 1$, and a handful of observations (x_i, y_i), which make the training data, suggest that the latent function varies a lot, then clearly one would need a number of basis functions considerably larger than the number of observations in order to capture such a variation. In effect, a handful number of basis functions would result in an almost constant optimal function, which wouldn't be appropriate. It is worth stressing that the issue here is not the choice of RKHS, which in this example is universal and consequently 'large' enough, but the issue is that the optimisation scheme does not explore the RKHS in a manner that is consistent with the evidence provided by the data, no matter g.

Favoring analytical tractability over empirical results is also very common in Bayesian nonparametric applications, where researchers almost systematically resort

to Gaussian process as priors over smooth functions for their nice analytical properties, and seldom contemplate the possibly that there could be alternatives. In doing so, researchers often forget that, when the optimal function is chosen to be the mean of the posterior process, as it is often the case in practice, the prior mean and covariance function of the functional prior almost entirely characterise the hypothesis space of candidate functions,[1] while the likelihood model and the dynamics of the functional prior (i.e. the nature of its finite-dimensional marginals) fully characterise how to explore the space of candidate functions, in the light of some training data, so as to find a suitable latent function. Hence, for a given likelihood/loss function, the learning question previously discussed is, in this case, essentially equivalent to *what stochastic process to choose as functional prior?* Clearly, in the same way we previously argued that the learning question plays an important role in deep learning and should play a more important role in frequentist kernel methods, the dynamics of the functional prior should also play a more important role in Bayesian kernel methods, and should not be chosen for analytical convenience.

To sum up, I strongly believe that, other than the issues of flexibility and scalability, which we hope we have addressed within this thesis, two of the main explanations for the success of deep learning relative to kernel learning are the amount of practical thoughts and engineering that have been put by deep learning researchers in answering the foregoing learning question, and their dedication to solving open practical AI challenges. This is something that I think the kernel learning community can learn from the deep learning community. I imagine a future where kernel learning researchers will acknowledge the importance of thinking beyond mainstream statistical formulations, such as regularised ERM and Gaussian process models, so that kernel methods may take their due place in solving open AI challenges.

[1] They do characterise it *exactly* when the RKHS induced by the prior covariance function is finite-dimensional.

References

Adams, R. P., Murray, I., and MacKay, D. (2009). Tractable nonparametric Bayesian inference in poisson processes with gaussian process intensities. pages 9–16.

Adams, R. P. and Stegle, O. (2008). Gaussian process product models for nonparametric nonstationarity. In Cohen, W. W., McCallum, A., and Roweis, S. T., editors, *ICML*, volume 307 of *ACM International Conference Proceeding Series*, pages 1–8. ACM.

Adler, R. J. and Taylor, J. E. (2011). *Topological Complexity of Smooth Random Functions: École D'Été de Probabilités de Saint-Flour XXXIX-2009*. Lecture Notes in Mathematics / École d'Été de Probabilités de Saint-Flour. Springer.

Agamennoni, G., Nieto, J. I., and Nebot, E. M. (2011). An outlier-robust kalman filter. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 1551–1558. IEEE.

Alquier, P., Friel, N., Everitt, R., and Boland, A. (2016). Noisy monte carlo: Convergence of markov chains with approximate transition kernels. *Statistics and Computing*, 26(1-2):29–47.

Bach, F. (2008). Exploring large feature spaces with hierarchical multiple kernel learning. *Advances in Neural Information Processing Systems (NIPS)*.

Bardenet, R., Doucet, A., and Holmes, C. (2014). Towards scaling up markov chain monte carlo: an adaptive subsampling approach. In *International Conference on Machine Learning (ICML)*, pages 405–413.

Basu, S. and Dassios, A. (2002). A cox process with log-normal intensity. *Insurance: mathematics and economics*, 31(2):297–302.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 35(8):1798–1828.

Bengio, Y., Lamblin, P., Popovici, D., and Larochelle, H. (2006). Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems*, pages 153–160.

Bergström, H. (2014). *Weak Convergence of Measures: Probability and Mathematical Statistics: A Series of Monographs and Textbooks*. Academic Press.

Bishop, C. (2007a). Conjugate Bayesian analysis of the Gaussian distribution. https://www.cs.ubc.ca/ murphyk/Papers/bayesGauss.pdf.

Bishop, C. (2007b). *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer.

Bishop, C. M. (1999). Bayesian PCA. In *Proceedings of the 1998 Conference on Advances in Neural Information Processing Systems II*, pages 382–388.

Bottou, L. (1998). Online learning and stochastic approximations. *Online Learning in Neural Networks*, 17(9):25.

Box, G. E. P., Jenkins, G. M., Reinsel, G. C., and M., L. G. (1970). *Time Series Analysis: Forecasting and Control*. Wiley.

Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Brooks, S., Gelman, A., Jones, G., and Meng, X.-L. (2011). *Handbook of Markov Chain Monte Carlo*. CRC press.

Cao, Y. and Fleet, D. (2014). Generalized product of experts for automatic and principled fusion of gaussian process predictions. *arXiv preprint arXiv:1410.7827*.

Chang, C.-C., Lee, Y., and Pao, H. (2010). A passive-aggressive algorithm for semi-supervised learning. In *International Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pages 335–341. IEEE.

Cho, Y. and Saul, L. K. (2009). Kernel methods for deep learning. In *Advances in Neural Information Processing Systems*, pages 342–350.

Commandeur, J. J. F. and Koopman, S. J. (2007). *State Space Time Series Analysis*. Oxford University Press.

Cornfeld, I. P., Fomin, S. V., and Sinai, Y. G. (2012). *Ergodic theory*, volume 245. Springer Science & Business Media.

Cox, D. (1955). Some statistical methods connected with series of events. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 129–164.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Cunningham, J., Byron, M., and Shenoy, K. (2007). Inferring neural firing rates from spike trains using gaussian processes. In *Advances in neural information processing systems*, pages 329–336.

Cunningham, J., Shenoy, K., and Sahani, M. (2008). Fast gaussian process methods for point process intensity estimation. In *Proceedings of the 25th international conference on Machine learning*, pages 192–199. ACM.

Daley, D. and Vere-Jones, D. (2008). *An Introduction to the Theory of Point Processes*. Springer-Verlag.

Damian, D., Sampson, P. D., and Guttorp, P. (2001). Bayesian estimation of semi-parametric nonstationary spatial covariance structures. *Environmetrics*, 12(2):161–178.

Damianou, A. C. and Lawrence, N. D. (2013). Deep gaussian processes. In *International Conference on Artificial Intelligence and Statistics*.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.

Dehay, D. and Moché, R. (1986). Strongly harmonizable approximations of bounded continuous random fields. *Stochastic processes and their applications*, 23(2):327–331.

Dehay, D. and Moché, R. (1992). Trace Measures of a Positive Definite Bimeasure. *Journal of Multivariate Analysis*, 40(1):115–131.

Deisenroth, M. and Ng, J. (2015). Distributed gaussian processes. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1481–1490.

Diggle, P. (1985). A kernel method for smoothing point process data. *Applied statistics*, pages 138–147.

Doob, J. L. (1944). The elementary gaussian processes. *The Annals of Mathematical Statistics*, 15(3):229–282.

Durbin, J. and Koopman, S. J. (2012). *Time Series Analysis by State Space Methods*. Oxford University Press.

Durrande, N., Ginsbourger, D., and Roustant, O. (2012). Additive covariance kernels for high-dimensional Gaussian process modeling. *Annales de la Faculté de Sciences de Toulouse*, 21(3).

Duvenaud, D., Lloyd, J. R., Grosse, R., B., T. J., and Ghahramani, Z. (2013). Structure discovery in nonparametric regression through compositional kernel search. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1166–1174.

Duvenaud, D., Nickisch, H., and Rasmussen, C. E. (2011). Additive Gaussian processes. In Shawe-Taylor, J., Zemel, R., Bartlett, P., Pereira, F., and Weinberger, K., editors, *Advances in Neural Information Processing Systems 24*, pages 226–234.

Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

Escobar, M. D. and West, M. (1995). Bayesian density estimation and inference using mixtures. *Journal of the american statistical association*, 90(430):577–588.

Fabec, R. C. (2000). *Fundamentals of Infinite Dimensional Representation Theory*. CRC Press.

Ferguson, T. (1973). A Bayesian analysis of some nonparametric problems. *The Annals of Statistics*, pages 209–230.

Foti, N. and Williamson, S. (2015). A survey of non-exchangeable priors for Bayesian nonparametric models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(2):359–371.

Gal, Y. and Turner, R. (2015). Improving the gaussian process sparse spectrum approximation by representing uncertainty in frequency inputs. In *International Conference on Machine Learning*.

Gelman, A., Carlin, J., Stern, H., Dunson, D., Vehtari, A., and Rubin, D., editors (2013). *Bayesian Data Analysis Thrid Edition*. CRC Press.

Geman, S. and Geman, D. (1984). Stochastic relaxation, gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741.

Genton, M. G. (2002). Classes of kernels for machine learning: A statistics perspective. *The Journal of Machine Learning Research*, 2:299–312.

Gönen, M. and Alpaydın, E. (2011). Multiple Kernel Learning Algorithms. *The Journal of Machine Learning Research*, 12:2211–2268.

Gramacy, R. B. and Lee, H. K. H. (2008). Bayesian treed Gaussian process models with an application to computer modeling. *Journal of the American Statistical Association*, 103(483).

Green, P. (1995). Reversible jump markov chain monte carlo computation and Bayesian model determination. *Biometrika*, 82(4):711–732.

Green, P. and Hastie, D. (2009). Reversible jump mcmc. *Genetics*, 155(3):1391–1403.

Gregory, P. and Loredo, T. (1992). A new method for the detection of a periodic signal of unknown shape and period. *The Astrophysical Journal*, 398:146–168.

Halmos, P. R. (1950). *Measure theory*. Springer-Verlag.

Hamilton, J. D. (1994). *Time series analysis*. Princeton University Press.

Hastings, W. (1970). Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 24:97–109.

Heikkinen, J. and Arjas, E. (1999). Modeling a poisson forest in variable elevations: a nonparametric Bayesian approach. *Biometrics*, 55(3):738–745.

Helmbold, D. P., Kivinen, J., and Warmuth, M. K. (1999). Relative loss bounds for single neurons. *Neural Networks, IEEE Transactions on*, 10(6):1291–1304.

Hensman, J., Fusi, N., and Lawrence, N. (2013). Gaussian processes for big data. In *Conference on Uncertainty in Artificial Intellegence*, pages 282–290. auai.org.

Hensman, J., Matthews, A., and Ghahramani, Z. (2015). Scalable variational gaussian process classification. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2015)*, JMLR Workshop and Conference Proceedings, pages 351–360.

Hildebrand, F. (1987). *Introduction to numerical analysis*. Courier Corporation.

Hinton, G. E., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554.

Hoffman, M. D., Blei, D. M., Wang, C., and Paisley, J. (2013). Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347.

Hofmann, T., Schölkopf, B., and Smola, A. J. (2008). Kernel methods in machine learning. *The annals of statistics*, pages 1171–1220.

Hsing, T. and Eubank, R. (2015). *Theoretical foundations of functional data analysis, with an introduction to linear operators*. John Wiley & Sons.

Hu, S. and Papageorgiou, N. (2013). *Handbook of Multivalued Analysis: Volume II: Applications*, volume 500. Springer Science & Business Media.

Jarrett, R. (1979). A note on the intervals between coal-mining disasters. *Biometrika*, 66(1):191–193.

Kakihara, Y. (1985). A note on harmonizable and v-bounded processes. *Journal of Multivariate Analysis*, 16:140–156.

Kakihara, Y. (1997). *Multidimensional second order stochastic processes*, volume 2. World Scientific.

Kallianpur, G. (1970). The role of reproducing kernel hilbert spaces in the study of gaussian processes. *Advances in probability and related topics*, 2:49–83.

Kalman, R. E. (1960). A new approach to linear filtering and prediction problems. *Journal of Fluids Engineering*, 82(1):35–45.

Karatzas, I. and Fernholz, R. (2009). Stochastic Portfolio Ttheory: An Overview. *Handbook of Numerical Analysis*, 15:89–167.

Kato, T. (2012). *Perturbation Theory for Linear Operators*. Springer Science & Business Media.

Kelley, H. J. (1960). Gradient theory of optimal flight paths. *Ars Journal*, 30(10):947–954.

Kim, H., K., M. B., and C., H. C. (2005). Analyzing nonstationary spatial data using piecewise Gaussian processes. *Journal of the American Statistical Association*, 100(470).

Kingman, J. (1967). Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78.

Kingman, J. F. C. (1993). *Poisson processes*. Wiley Online Library.

Kivinen, J. and Warmuth, M. K. (1997). Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1–63.

Kottas, A. (2006). Dirichlet process mixtures of beta distributions, with applications to density and intensity estimation. In *Workshop on Learning with Nonparametric Bayesian Methods, 23rd International Conference on Machine Learning (ICML)*. Citeseer.

Kottas, A. and Sansó, B. (2007). Bayesian mixture modeling for spatial poisson process intensities, with applications to extreme value analysis. *Journal of Statistical Planning and Inference*, 137(10):3151–3163.

Kwiatkowski, D., Phillips, P. C. B., Schmidt, P., and Shin, Y. (1992). Testing the null hypothesis of stationarity against the alternative of a unit root: How sure are we that economic time series have a unit root? *Journal of econometrics*, 54(1):159–178.

Lawrence, N., Seeger, M., and Herbrich, R. (2003). Fast sparse Gaussian process methods: The informative vector machine. In *Advances in Neural Information Processing Systems 15*, pages 625–632. MIT Press.

Lawrence, N. D. (2003). Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems*.

Lazaro-Gredilla, M., Quinonero-Candela, J., Rasmussen, C. E., and Figueiras-Vida, A. R. (2010). Sparse spectrum gaussian process regression. *The Journal of Machine Learning Research*, 11:1866–1881.

Le, Q., Sarlós, T., and Smola, A. (2013). Fastfood-approximating kernel expansions in loglinear time. In *International Conference on Machine Learning (ICML)*.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.

Littlestone, N. (1989). Mistake bounds and logarithmic linear-threshold learning algorithms.

Loève, M. (1963). *Probability Theory II (Graduate Texts in Mathematics)*, volume 2. Springer-Verlag.

Macdonald, I. (1995). *Symmetric Functions and Hall Polynomials*. Oxford University Press.

MacKay, D. J. C. (1998). Introduction to Gaussian processes. In Bishop, C. M., editor, *Neural Networks and Machine Learning, volume 168 of NATO ASI Series*, pages 133–165. Springer, Berlin.

Meeds, E. and Osindero, S. (2006). An alternative infinite mixture of Gaussian process experts. In *Advances In Neural Information Processing Systems*.

Mehlman, M. H. (2004). Some properties of harmonizable processes. *Advances on Theoretical and Methodological Aspects of Probability and Statistics*, page 49.

Micchelli, C., Xu, Y., and Zhang, H. (2006). Universal Kernels. *The Journal of Machine Learning Research*, 7:2651–2667.

Močkus, J. (1975). On Bayesian methods for seeking the extremum. In *Optimization Techniques IFIP Technical Conference*, pages 400–404. Springer.

Močkus, J. (2012). *Bayesian approach to global optimization: theory and applications*, volume 37. Springer Science & Business Media.

Moeller, J., Syversveen, A., and Waagepetersen, R. (1998). Log-gaussian cox processes. *Scandinavian Journal of Statistics*.

Moeller, J., Syversveen, A., and Waagepetersen, R., editors (2003). *Statistial Inference and Simulation for Spatial Point Processes*. Chapman Hall/CRC.

Morse, M. (1955). Bimeasures and their integral extensions. *Annali di Matematica Pura ed Applicata*, 39(1):345–356.

Murphy, K. P. (2012). *Machine learning: a probabilistic perspective*. MIT press.

Murray, I., Adams, R. P., and MacKay, D. (2010). Elliptical slice sampling. pages 9–16. Appearing in Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS) 2010.

Neal, R. M. (1996). *Bayesian learning for neural networks*. Lecture notes in statistics. Springer.

Neal, R. M. (2000). Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265.

Øksendal, B. (2003). *Stochastic Differential Equations: An Introduction with Applications*. Hochschultext / Universitext. Springer.

Oliva, J., Dubey, A., Poczos, B., Schneider, J., and Xing, E. (2016). Bayesian nonparametric kernel-learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*.

Paciorek, C. and Schervish, M. (2004). Nonstationary covariance functions for Gaussian process regression. *Advances in neural information processing systems*, 16:273–280.

Papoulis, A. and Pillai, S. U. (2002). *Probability, Random Variables, and Stochastic Processes*. Tata McGraw-Hill Education.

Pitman, J. and Yor, M. (1997). The two-parameter poisson-dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pages 855–900.

Plagemann, C., Kersting, K., and Burgard, W. (2008). Nonstationary Gaussian process regression using point estimates of local smoothness. In *Proc. of the European Conference on Machine Learning (ECML)*.

Poultney, C., Chopra, S., and LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144.

Quinonero-Candela, J. and Rasmussen, C. (2005). A unifying view of sparse approximate gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1177–1184.

Rao, V. A. and Teh, Y. W. (2011). Gaussian process modulated renewal processes. *Neural Information Processing Systems (NIPS)*.

Rasmussen, C. E. and Ghahramani, Z. (2001). Infinite mixtures of Gaussian process experts. In *Advances in Neural Information Processing Systems 14*, pages 881–888. MIT Press.

Rasmussen, C. E. and Williams, C. K. I. (2005). *Gaussian Processes for Machine Learning*. The MIT Press.

Rathbum, S. and Cressie, N. (1994). Asymptotic properties of estimators for the parameters of spatial inhomogeneous poisson point processes. *Advances in Applied Probability*, 26:122–154.

Ross, J. and Dy, J. (2013). Nonparametric mixture of Gaussian processes with constraints. In Dasgupta, S. and Mcallester, D., editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 1346–1354. JMLR Workshop and Conference Proceedings.

Rudin, W. (1962). *Fourier analysis on groups*. John Wiley.

Rumelhart, D. E., Hinton, G. E., and Williams, D. R. (1986). Learning representations by back-propagating errors. *Nature*, 323:533–536.

Saatchi, Y. (2011). *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge.

Sampson, P. D. and Guttorp, P. (1992). Nonparametric estimation of nonstationary spatial covariance structure. *Journal of the American Statistical Association*, 87(417).

Särkkä, S., Solin, A., and Hartikainen, J. (2013). Spatiotemporal learning via infinite-dimensional Bayesian filtering and smoothing: A look at gaussian process regression through kalman filtering. *IEEE Signal Process. Mag.*, 30(4):51–61.

Schmidt, A. M. and O'Hagan, A. (2003). Bayesian inference for nonstationary spatial covariance structure via spatial deformations. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 65(3):743–758.

Schölkopf, B. and Smola, A. (2001). *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, Cambridge, MA, USA.

Schölkopf, B., Smola, A., and Müller, K.-R. (1997). Kernel Principal Component Analysis. In *Artificial Neural Networks—ICANN'97*, pages 583–588. Springer.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

Seeger, M. (2003a). Bayesian Gaussian process models: Pac-Bayesian generalisation error bounds and sparse approximations. Technical report.

Seeger, M. (2003b). Pac-bayesian generalisation error bounds for Gaussian process classification. *The Journal of Machine Learning Research*, 3:233–269.

Shah, A., Wilson, A. G., and Ghahramani, Z. (2014). Student-t processes as alternatives to gaussian processes. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS 2014)*, JMLR Workshop and Conference Proceedings, pages 877–885.

Shi, T. and Zhu, J. (2014). Online Bayesian passive-aggressive learning. In *International Conference on Machine Learning (ICML)*, pages 378–386.

Silverman, B. W. (1985). Some Aspects of the Spline Smoothing Approach to Non-Parametric Regression Curve Fitting. *Journal of the Royal Statistical Society. Series B (Methodological)*, 47(1):1–52.

Silverman, R. (1957). Locally stationary random processes. *Information Theory, IRE Transactions on*, 3(3):182–187.

Smola, A. J. and Bartlett, P. (2001). Sparse greedy Gaussian process regression. In *Advances in Neural Information Processing Systems 13*, pages 619–625. MIT Press.

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In *Advances in Neural Information Processing Systems*, pages 1257–1264. MIT press.

Solin, A. and Särkkä, S. (2014). Explicit link between periodic covariance functions and state space models. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Statistics (AISTATS 2014)*, volume 33 of *JMLR Workshop and Conference Proceedings*, pages 904–912.

Sonnenburg, S., Rätsch, G., Schäfer, C., and Schölkopf, B. (2006). Large Scale Multiple Kernel Learning. *The Journal of Machine Learning Research*, 7:1531–1565.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

Stein, M. L. (1999). *Interpolation of Spatial Data*. Springer-Verlag, New York.

Steinwart, I. and Christmann, A. (2008). *Support Vector Machines*. Springer Science & Business Media.

Steinwart, I., Hush, D., and Scovel, C. (2006). Function classes that approximate the Bayes risk. In *Learning Theory*, pages 79–93. Springer.

Sundaram, R. (1996). *A first course in optimization theory*. Cambridge university press.

Teh, Y. W. (2011). Dirichlet process. In *Encyclopedia of machine learning*, pages 280–287. Springer.

The GPy authors (2012–2016). GPy: A gaussian process framework in python. http://github.com/SheffieldML/GPy.

Tresp, V. (2000). A Bayesian Committee Machine. *Neural Computation*, 12(11):2719–2741.

Tresp, V. (2001). Mixtures of Gaussian processes. In *Advances in Neural Information Processing Systems 13*, pages 654–660. MIT Press.

Twitter (2014). Twitter Sample Stream API.

Vervuurt, A. and Karatzas, I. (2015). Diversity-weighted portfolios with negative parameter. *Annals of Finance*, 11(3):411–432.

Wang, Z. and Vucetic, S. (2010). Online passive-aggressive algorithms on a budget. In *International Conference on Artificial Intelligence and Statistics*, pages 908–915.

West, M., Harrison, P. J., and Migon, H. S. (1985). Dynamic generalized linear models and Bayesian forecasting. *Journal of the American Statistical Association*, 80(389):73–83.

Williams, C. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press.

Williams, D. (1991). *Probability with martingales*. Cambridge university press.

Wilson, A. and Nickisch, H. (2015). Kernel interpolation for scalable structured gaussian processes (kiss-gp). In *Proceedings of The 32nd International Conference on Machine Learning*, pages 1775–1784.

Wilson, A. G. and Adams, R. P. (2013). Gaussian process kernels for pattern discovery and extrapolation. In *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, pages 1067–1075.

Wilson, A. G., Gilboa, E., and Nehorai, A.and Cunningham, J. P. (2014). Fast kernel learning for multidimensional pattern extrapolation. In *Advances in Neural Information Processing Systems*. MIT Press.

Wilson, A. G., Hu, Z., Salakhutdinov, R., and Xing, E. P. (2016). Deep kernel learning. In *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*.

Yaglom, A. M. (1987). *Correlation theory of stationary and related random functions*, volume 1. Springer-Verlag.

Yang, Z., Smola, A., Song, L., and Wilson, A. G. (2015). A la carte – learning fast kernels. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, pages 1098–1106.

Appendix A

Derivations of Chapter 2

A.1 The Poisson Process Likelihood is Weakly Informative

In this section we prove the proposition below.

Proposition A.1 *Let $\mathbb{Q}^*$ be an $(n+1)$-dimensional continuous probability distribution whose density has support $\bigotimes_{i=1}^{n+1}]0,+\infty[$, and let $x_1,\ldots,x_n$ be n points on a compact domain $\mathcal{S}\subset\mathbb{R}^d$. There exists an almost surely non-negative and $\mathcal{C}^\infty$ stochastic process λ on $\mathcal{S}$ such that*

$$\left(\lambda^*(x_1),...,\lambda^*(x_n),\int_{\mathcal{S}}\lambda^*(x)dx\right)\sim\mathbb{Q}^*.$$

Proof Let

$$(y_1,\ldots,y_n,I)\sim\mathbb{Q}^*$$

and

$$(y_1(\omega),\ldots,y_n(\omega),I(\omega))$$

a random draw. Let us denote $x[j], j\leq d$ the j-th coordinate of $x\in\mathbb{R}^d$. We consider the family of functions parametrized by $\alpha\in\mathbb{R}$:

$$f(\omega,x,\alpha)=\exp\left(\alpha\sum_{j=1}^{d}\prod_{l=1}^{n}(x[j]-x_l[j])^2\right)\sum_{l=1}^{n}y_l(\omega)\frac{1}{d}\sum_{j=1}^{d}\prod_{k\neq l}\left(\frac{x[j]-x_k[j]}{x_l[j]-x_k[j]}\right)^2. \quad \text{(A.1)}$$

We note that $\forall\alpha,x_i,\ f(\omega,x_i,\alpha)=y_i(\omega)$. Let us define the random polynomial

$$P(x)=\sum_{l=1}^{n}y_l(\omega)\frac{1}{d}\sum_{j=1}^{d}\prod_{k\neq l}\left(\frac{x[j]-x_k[j]}{x_l[j]-x_k[j]}\right)^2.$$

As P is continuous, it is bounded on the compact $\mathcal{S}$, and reaches its bounds. Thus we have

$$\exists\ m_p, M_p \geq 0,\ \text{s.t.}\ \forall x \in \mathcal{S},\ 0 \leq m_p \leq P(x) \leq M_p.$$

Similarly, if we define

$$R(\alpha, x) = \exp\left(\alpha \sum_{j=1}^{d} \prod_{l=1}^{n} (x[j] - x_l[j])^2\right) = R(1, x)^\alpha,$$

it follows that

$$\exists\ m_q, M_q > 1,\ \text{s.t.}\ \forall x \in \mathcal{S},\ 1 < m_q \leq R(1, x) \leq M_q.$$

Hence,

$$m_p m_q^\alpha \mu(\mathcal{S}) \leq \int_{\mathcal{S}} f(\omega, x, \alpha) dx \leq M_p M_q^\alpha \mu(\mathcal{S}). \tag{A.2}$$

Moreover, we note that $\alpha \to \int_{\mathcal{S}} f(\omega, x, \alpha) dx$ is continuous on $\mathbb{R}$ as its restriction to any bounded interval is continuous (by dominated convergence theorem). Furthermore, given that $m_q, M_q > 1$, it follows from Equation (A.2) that

$$\lim_{\alpha \to +\infty} \int_{\mathcal{S}} f(\omega, x, \alpha) dx = +\infty$$

and

$$\lim_{\alpha \to -\infty} \int_{\mathcal{S}} f(\omega, x, \alpha) dx = 0.$$

Hence, by intermediate value theorem,

$$\forall\ I(\omega) > 0,\ \exists \alpha^*(\omega)\ \text{s.t.}\ I(\omega) = \int_{\mathcal{S}} f(\omega, x, \alpha^*(\omega)) dx.$$

Finally, let us define the stochastic process λ on $\mathcal{S}$ as

$$\omega \to \lambda(\omega, x) := f(\omega, x, \alpha^*(\omega)).$$

To summarise,

$$\forall\ x_i, \lambda^*(\omega, x_i) := f(\omega, x_i, \alpha^*(\omega)) = y_i(\omega),$$
$$I(\omega) = \int_{\mathcal{S}} \lambda^*(\omega, x) dx,$$

and

$$(y_1, \ldots, y_n, I) \sim \mathbb{Q} :$$

this implies $\left(\lambda^*(x_1), ..., \lambda^*(x_n), \int_{\mathcal{S}} \lambda^*(x)dx\right) \sim \mathbb{Q}$. Finally,

$$\forall\ x \in \mathcal{S},\ \lambda^*(\omega, x) \geq 0,\ \text{and}\ \forall\ \omega,\ x \to \lambda^*(\omega, x)\ \text{is}\ \mathcal{C}^\infty,$$

which concludes our proof. ■

A.2 Proof of Convergence of Algorithm 2.1

The idea behind this proof is to show that the sequence of maximum utility

$$u_k = \max_{s \in \mathcal{S}} \tilde{\mathcal{U}}(\{s'_1, ..., s'_{k-1}\} \cup \{s\})$$

is positive, increasing and upper-bounded and thus converges to a strictly positive limit. This would then imply that

$$\frac{u_{k+1} - u_k}{u_k} \underset{k \to \infty}{\longrightarrow} 0$$

and subsequently that

$$\forall\ 0 < \alpha < 1, \exists\ k_{\text{lim}} \in \mathbb{N}\ \text{s.t.}\ \forall\ k > k_{\text{lim}}, \frac{u_{k+1} - u_k}{u_k} < \alpha$$

or in other words Algorithm 2.1 always stops in finite time.

To show that $\forall k > 0,\ u_k > 0$, we note that $\Sigma^*_{\mathcal{D}'\mathcal{D}'}(\tilde{\theta}_i)$ is the covariance matrix of a non-degenerate Gaussian vector and as such it is positive definite. It follows that $\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}(\tilde{\theta}_i)$ is also positive definite. We further note that the j-th diagonal term of $\Sigma^*_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*T}_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)$ can be written as $x_j^T\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}(\tilde{\theta}_i)x_j$ where x_j is the j-th column of $\Sigma^{*T}_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)$. Hence, by virtue of the positive definitiveness of $\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}(\tilde{\theta}_i)$, the diagonal terms of $\Sigma^*_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*T}_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)$ are all positive, which proves that the utility function $\tilde{\mathcal{U}}$ is positive, and subsequently that $\forall k > 0, u_k > 0$. To show that $(u_k)_{k \in \mathbb{N}^*}$ is upper-bounded, we note that the matrix

$$C_{i\mathcal{D}'} = \Sigma^*_{\mathcal{D}\mathcal{D}}(\tilde{\theta}_i) - \Sigma^*_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*T}_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)$$

where the notation is as per Chapter 2, is an auto-covariance matrix, and as such has positive diagonal elements. Hence,

$$\mathrm{Tr}(\Sigma^*_{\mathcal{D}\mathcal{D}}(\tilde{\theta}_i)) \geq \mathrm{Tr}(\Sigma^*_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*-1}_{\mathcal{D}'\mathcal{D}'}(\tilde{\theta}_i)\Sigma^{*T}_{\mathcal{D}\mathcal{D}'}(\tilde{\theta}_i))$$

and finally

$$\forall\ k \in \mathbb{N}^*, u_k \leq \frac{1}{N}\sum_{i=1}^{N} \mathrm{Tr}\left(\Sigma^*_{\mathcal{D}\mathcal{D}}(\tilde{\theta}_i)\right).$$

Moreover, we note that showing that $(u_k)_{k\in\mathbb{N}^*}$ is increasing is equivalent to showing that $(v_k)_{k\in\mathbb{N}^*}$ with

$$v_k = \min_{s\in\mathcal{S}} \frac{1}{N}\sum_{i=1}^{N} \mathrm{Tr}(C_{i\{s'_1,\dots,s'_{k-1}\}\cup\{s\}})$$

is decreasing. We recall that $C_{i\{s'_1,\dots,s'_{k-1}\}\cup\{s\}}$ is the covariance matrix of the values of the stationary Gaussian process of our model at the data points, conditioned on its values at $\{s'_1,\dots,s'_{k-1}\}\cup\{s\}$. It follows from the law of iterated expectations that $C_{i\{s'_1,\dots,s'_{k-1}\}\cup\{s\}}$ could also be seen as the covariance matrix of the values of a conditional Gaussian process at the data points[1], conditioned on its value at s. Hence,

$$C_{i\{s'_1,\dots,s'_{k-1}\}\cup\{s\}} = C_{i\{s'_1,\dots,s'_{k-1}\}} - \frac{1}{\hat{\Sigma}_{ss}(\tilde{\theta}_i)}\hat{\Sigma}_{\mathcal{D}\{s\}}(\tilde{\theta}_i)\hat{\Sigma}^T_{\mathcal{D}\{s\}}(\tilde{\theta}_i)$$

where $\hat{\Sigma}_{XY}$ denotes the covariance matrix between the values of the conditional GP at points in X and at points in Y. In particular, $\hat{\Sigma}_{ss}(\tilde{\theta}_i)$ is a positive scalar. What's more the diagonal elements of $\hat{\Sigma}_{\mathcal{D}\{s\}}(\tilde{\theta}_i)\hat{\Sigma}^T_{\mathcal{D}\{s\}}(\tilde{\theta}_i)$ are all non-negative. Hence,

$$\forall s \in \mathcal{S}, \mathrm{Tr}(C_{i\{s'_1,\dots,s'_{k-1}\}\cup\{s\}}) \leq \mathrm{Tr}(C_{i\{s'_1,\dots,s'_{k-1}\}})$$

and averaging over the set of hyper-parameters θ_i and taking the min we get

$$\forall k \geq 2,\ v_k \leq v_{k-1}$$

which concludes the proof.

[1]The conditional GP is defined as the stationary Gaussian process in our model that is conditioned on its values at the points $\{s'_1,\dots,s'_{k-1}\}$.

A.3 Proof of Rate of Convergence of Algorithm 2.1

The key idea of this proof is to note as previously shown that no set of inducing points has a utility greater than $w_\infty := \frac{1}{N}\sum_{i=1}^{N} \mathrm{Tr}(\Sigma^*_{\mathcal{D}\mathcal{D}}(\tilde{\theta}_i))$, but that any set of inducing points that includes $\mathcal{D}$ has a utility equal to w_∞.

Let $\{s'_1, ..., s'_k\}$ be points selected after k iterations of Algorithm 2.1, and let us denote by $\{u_1, ..., u_k\}$ the maximum utilities after the corresponding iterations as usual. Let us denote by

$$\tilde{s}_k = \underset{s \in \mathcal{D}}{\mathrm{argmax}}\, \tilde{\mathcal{U}}(\{s'_1, ..., s'_{k-1}\} \cup \{s\})$$

the best candidate *in the data set* to be the k-th inducing point after $k-1$ iterations of our algorithm. As previously mentioned, $\{s'_1, ..., s'_{k-1}\} \cup \mathcal{D}$ is a set of inducing points with perfect utility. Therefore, if we select the data points as inducing points after $\{s'_1, ..., s'_{k-1}\}$, their contribution to the overall utility will be $w_\infty - u_{k-1}$. If we further constrain our choice of $\mathcal{D}$ as additional inducing points to start with $\tilde{s}_k$ then the incremental utility of choosing $\tilde{s}_k$ will be at least $\frac{w_\infty - u_{k-1}}{n}$, where n is the data size as usual. This is because $\tilde{s}_k$ is the best choice for the k-th inducing point in $\mathcal{D}$ after having picked $\{s'_1, ..., s'_{k-1}\}$ and because the incremental utility of choosing an inducing point is higher earlier (when little is known about the GP) than later (when more is known about the GP). What's more, by definition, the incremental utility of choosing s'_k after $\{s'_1, ..., s'_{k-1}\}$ is higher than that of choosing $\tilde{s}_k$ after $\{s'_1, ..., s'_{k-1}\}$. Hence,

$$u_k - u_{k-1} \geq \frac{w_\infty - u_{k-1}}{n}.$$

Let us denote by w_k the sequence satisfying

$$w_0 = u_0, \forall\ k \in \mathbb{N}^* w_k - w_{k-1} = \frac{w_\infty - w_{k-1}}{n}.$$

It can be shown (by induction on k) that

$$\forall\ k \in \mathbb{N}^* w_k \leq u_k.$$

Moreover, we note that

$$w_k - w_\infty = (1 - \frac{1}{n})(w_{k-1} - w_\infty).$$

Hence

$$w_k = w_\infty + (1 - \frac{1}{n})^k (w_0 - w_\infty),$$

which proves that the sequence w_k converges linearly to w_∞ with rate $1 - \frac{1}{n}$. On one hand, we have shown that the sequence u_k converges and is upper-bounded by w_∞, hence its limit is smaller than w_∞:

$$u_\infty := \lim_{k \to \infty} u_k \leq w_\infty.$$

On the other hand, we have shown that $\forall\ k \in \mathbb{N}^*\ w_k \leq u_k$ which implies

$$w_\infty \leq u_\infty.$$

Hence,

$$w_\infty = u_\infty = \frac{1}{N} \sum_{i=1}^{N} \mathrm{Tr}(\Sigma^*_{\mathcal{DD}}(\tilde{\theta}_i)).$$

As w_k is upper-bounded by u_k and both sequences converge to the same limit, u_k, and subsequently Algorithm 2.1, converge at least as fast as w_k. In regards to the second statement of our proposition, we have that

$$\lim_{\alpha \to 0} u_f(\alpha) = \lim_{k \to \infty} u_k = \frac{1}{N} \sum_{i=1}^{N} \mathrm{Tr}(\Sigma^*_{\mathcal{DD}}(\tilde{\theta}_i)).$$

Appendix B

Derivations of Chapter 3

Unless stated otherwise, the stochastic processes we consider throughout this appendix are indexed in $\mathbb{R}^+$. To ease notations, we use the superscript $^{(i)}$ to denote the i-th order derivative of a function or stochastic process when it exists, or the original function or stochastic process when $i = 0$. In the case of stochastic processes, the derivative is to be understood in the mean square sense. Moreover, stationarity of stochastic processes is always meant in the weak sense. Furthermore, observation times are always assumed to be distinct and sorted by index: $t_0 < \cdots < t_k < \ldots$.

B.1 Discussion on the Trend-Stationary Gaussian Process Assumption

Objective: In this section we discuss the appropriateness of assuming that the latent time series is a trend-stationary Gaussian process. More precisely, we argue that given *a single path* of a time series *on some bounded interval* $[0, T]$, neither the trend-stationarity assumption nor the Gaussianity assumption can be invalidated.

Trend-Stationarity: Firstly, we note that unless further assumptions are made about the trend other than it being smooth, trend-stationarity cannot be invalidated using discrete observations of a *single path* as we can always find an infinitely smooth trend or mean function $\hat{m}$ (using polynomials for instance), that coincides with all discrete observations, making the observed path highly likely to result from any trend-stationary stochastic process with mean function $\hat{m}$.

In practice however, the trend might be assumed to lie in a parametric family. Even in that case, the stationarity of the residual time series can hardly be invalidated.

In effect, most stationarity statistical tests have as null hypothesis that the sample comes from a nonstationary time series. Evidence is then gathered from the sample through the test statistics to determine whether the null hypothesis can be rejected with some confidence, or equivalently if there is enough evidence in the sample to conclude stationarity. Hence, fundamentally, such an approach cannot be used to conclude nonstationarity, as the lack of evidence of stationarity in a *given sample* is not an evidence of nonstationarity of the *underlying process*. It might well be that the sample does not contain enough information to fully characterise the underlying stochastic process so that, had we collected more data, we might have been able to conclude stationarity. It is in the same spirit that Kwiatkowski et al. (1992) noted that 'most economic time series are not very informative about whether or not there is a unit root'. As we do not assume that we have enough data to characterize the latent process, our trend-stationarity assumption cannot be invalidated experimentally.

To illustrate our point, we drew a path from a stationary Gaussian process with mean 0 and squared exponential covariance function $k(u, v) = e^{-\frac{1}{2}(u-v)^2}$ on $[0, 10]$ discretized with mesh size 0.001 (see Figure B.1). We ran three standard stationarity

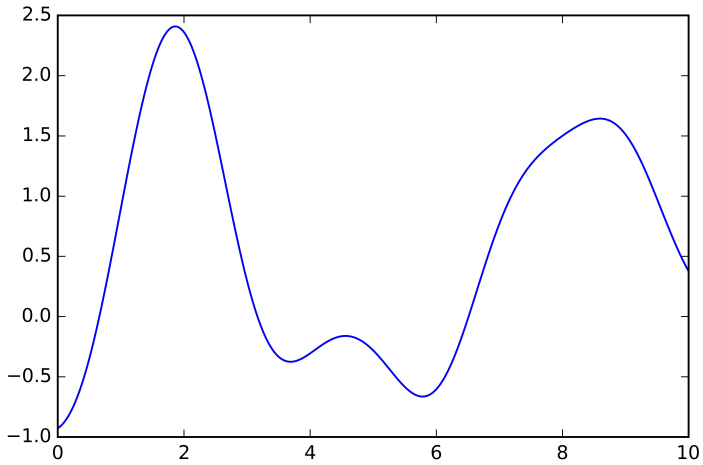

Fig. B.1 Draw from a stationary GP on $[0, 10]$.

statistical tests on the sample, namely the Augmented Dickey-Fuller test (ADF), the ADF-GLS test and the Phillips-Perron test. As can be seen in Table B.1, all three tests failed to find evidence for (to conclude) stationarity with 80% confidence, despite the sample coming from a stationary stochastic process. As all three stationarity tests rely on ergodicity for the first two moments, one might be tempted to think that this could be an indication that the underlying stochastic process is not ergodic. However,

Table B.1 Results of stationarity tests on the time series in Figure B.1.

Test	Statistics	p-Value	Lags
ADF	-2.12	0.24	22
ADF-GLS	-1.02	0.28	22
Phillips-Perron	-1.64	0.46	22

a sufficient condition for a mean zero stationary Gaussian process to be ergodic for the first two moments is that its covariance function vanishes as the lag increases (see Papoulis and Pillai, 2002, §13.1)

$$\forall t,\ k(\tau) := \mathrm{cov}(z_t, z_{t+\tau}) \underset{\tau \to +\infty}{\longrightarrow} 0,$$

and this condition is satisfied by the squared exponential kernel. The real issue at play here is that, given a *finite sample* of a time series, it is hardly possible to say with confidence that it comes from a nonstationary time series.

Gaussianity: Similarly, with one single realisation, it is hardly possible to determine whether the sample comes from a multivariate Gaussian, and thus the Gaussian process assumption cannot be invalidated either. Testing whether a real-valued vector is a draw from a multivariate Gaussian without any assumption on its mean and covariance matrix is as hopeless as testing whether a real-value scalar is a draw from a univariate Gaussian random variable without any assumption on its mean or variance.

B.2 Mean Square Differentiability and Markovianity Implies a Constant Covariance Function

Objective: In this section we prove that if a real-valued trend-stationary Gaussian process with a differentiable mean function is mean square differentiable and Markovian, then it has a constant covariance function (or equivalently it is almost surely equal to its mean function plus a constant).

Proof Let us consider a real-valued trend-stationary, mean square differentiable and Markovian Gaussian process $(z_t)_{t\geq 0}$. Let us denote $(u, v) \to k(u - v)$ its covariance

function. It follows that

$$\forall t, h,\ \operatorname{cov}\left(z_{t+h}, z_{t-h} | z_t\right) = k(2h) - \frac{k(h)^2}{k(0)} = 0, \tag{B.1}$$

where the first equality results from standard Gaussian identities and the second equality results from $(z_t)_{t\geq 0}$ being Markovian. As $(z_t)_{t\geq 0}$ is also assumed to be mean square differentiable and to have a differentiable mean function, the centred Gaussian process $(z_t - \mathrm{E}(z_t))_{t\geq 0}$ is mean square differentiable, or equivalently k is twice differentiable everywhere. Hence, $h \to k(2h) - \frac{k(h)^2}{k(0)}$ is also twice differentiable at 0 and has second order derivative $2k^{(2)}(0)$ at 0. It then follows from Equation (B.1) that $k^{(2)}(0) = 0$. As $h \to -k^{(2)}(h)$ is the covariance function of $(z_t^{(1)})_{t\geq 0}$, we have:

$$k^{(2)}(h)^2 = \operatorname{cov}\left(z_t^{(1)}, z_{t+h}^{(1)}\right)^2 \leq \operatorname{var}\left(z_t^{(1)}\right) \operatorname{var}\left(z_{t+h}^{(1)}\right) = k^{(2)}\left(0\right)^2 = 0.$$

That is, $\forall h, k^{(2)}\left(h\right) = 0$, or equivalently k is a linear function of h. Moreover, as $(z_t)_{t\geq 0}$ is trend-stationary k is also bounded[1]. Hence, k is a constant function. This means that for any times u and v, $z_u - \mathrm{E}(z_u)$ and $z_v - \mathrm{E}(z_v)$ have the same mean, the same variance and correlation 1, which implies

$$\forall u, v \geq 0,\ z_u - \mathrm{E}(z_u) \overset{\text{a.s.}}{=} z_v - \mathrm{E}(z_v).$$

■

B.3 Proof of Proposition 3.8

Let us prove Proposition 3.8, which we recall below.

Proposition: For any $\nu > 0$, the family of kernels we refer to as spectral Matérn kernels, which we define as

$$\left\{k_{\mathrm{SMA}}(\tau; \nu, n) : \omega_i \in \mathbb{R},\ k_{0i}, l_i \geq 0,\ n \in \mathbb{N}\right\},$$

with

$$k_{\mathrm{SMA}}(\tau; \nu, n) := \sum_{i=0}^{n} k_{\mathrm{MA}}(\tau; k_{0i}, l_i, \nu) \cos\left(2\pi\omega_i \tau\right),$$

[1] More precisely, the positive-definiteness of the covariance matrix between z_t and z_{t+h} implies $\forall h, |k(h)| \leq k(0)$.

where k_{MA} is the Matérn kernel defined in Equation (3.7), is pointwise dense in the family of continuous stationary covariance functions defined on $\mathbb{R}$.

Proof The family $\{k_{\text{SS}}(\tau; n) : \omega_i \in \mathbb{R}, k_{0i} \geq 0, n \in \mathbb{N}\}$ with k_{SS} defined in Equation (3.12), which we have already proved is pointwise dense in the family of all continuous stationary covariance functions, is obtained from spectral Matérn kernels as the limit case $l_i \to +\infty$. Hence the family

$$\{k_{\text{SMA}}(\tau; \nu, n) : \omega_i \in \mathbb{R},\ k_{0i}, l_i \geq 0,\ n \in \mathbb{N}\}$$

is also pointwise dense in the family of all continuous stationary covariance functions. ■

B.4 Proof of Proposition 3.9

Let us prove Proposition 3.9, which we recall below.

Proposition: Let $p(\mathcal{D}|k^*, m)$ denote the marginal likelihood in a GPR problem with training data $\mathcal{D}$, mean function m, and covariance function k^*. Then for any continuous stationary kernel k and for any $\nu > 0$, there exists a sequence of spectral Matérn kernels $\{k_{\text{SMA}}(\tau; \nu, n)\}_{n\in\mathbb{N}}$ such that

$$p\left(\mathcal{D}|k_{\text{SMA}}(.; \nu, n), m\right) \underset{n\to+\infty}{\longrightarrow} p(\mathcal{D}|k, m).$$

Proof Let $\{k_{\text{SMA}}(\tau; \nu, n)\}_{n\in\mathbb{N}}$ be a sequence of spectral Matérn kernels that converges pointwise to k. We may always find such a sequence because of Proposition 3.8. Let $\mathbf{f}$ denote the values of the latent GP at all training inputs. First we note that the multivariate Gaussian prior $p(\mathbf{f}|k^*, m)$ depends on the covariance function k^* only as a continuous function of the covariance/Gram matrix

$$[k^*(x_i, x_j)]_{i,j}$$

over all training inputs. Hence, pointwise convergence of the spectral Matérn kernels to k implies that the densities $p\left(\mathbf{f}|k_{\text{SMA}}(.; \nu, n), m\right)$ converge pointwise to $p(\mathbf{f}|k, m)$ as n goes to $+\infty$, which implies in particular that the probability measures $p\left(\mathbf{f}|k_{\text{SMA}}(.; \nu, n), m\right) d\mathbf{f}$ converge weakly to $p(\mathbf{f}|k, m)d\mathbf{f}$. Noting that the Gaussian likelihood $p(\mathcal{D}|\mathbf{f})$ is a continuous and bounded function of $\mathbf{f}$, a direct application of the definition of weak

convergence of measures gives us that

$$p\left(\mathcal{D}|k_{\text{SMA}}(.;\nu,n),m\right) = \int p(\mathcal{D}|\mathbf{f})p\left(\mathbf{f}|k_{\text{SMA}}(.;\nu,n),m\right)d\mathbf{f}$$

converges to

$$p\left(\mathcal{D}|k,m\right) = \int p(\mathcal{D}|\mathbf{f})p\left(\mathbf{f}|k,m\right)d\mathbf{f}$$

as n goes to $+\infty$. ■

B.5 State Space Representation of a Trend Stationary Gaussian Process with Matérn Covariance Function

Objective: In this section we derive the state space representation of the Gaussian process regression model under a trend-stationary latent GP $(z_t)_{t\geq 0}$, with mean function m, Matérn-$(p+\frac{1}{2})$ kernel, and with a Gaussian white noise $(\epsilon_t)_{t\geq 0}$.

Derivation: If we denote $f_t = z_t - m(t)$, $(x_t)_{t\geq 0}$ with $x_t = \left(f_t, f_t^{(1)}, \ldots, f_t^{(p)}\right)$ the pDGP corresponding to $(f_t)_{t\geq 0}$, and $H = (1, 0, \ldots, 0)$, it is easy to see that if we use x_t as state variable, the measurement equation of the state space model should be

$$\forall t,\ y_t = m(t) + H^T x_t + \epsilon_t.$$

Denoting $K_{u,v}$ the cross-covariance matrix between x_u and x_v, $K_{u|v} = L_{u|v}L_{u|v}^T$ the auto-covariance matrix of x_u conditional on x_v, and t_0 the initial time, we get the initial state distribution:

$$x_{t_0} \sim \mathcal{N}(0, K_{t_0,t_0}).$$

For observations times $t_0, \ldots, t_T$, using Bayes' rule and Corollary 3.5, we get

$$p\left(x_{t_0}, \ldots, x_{t_T}\right) = p(x_{t_0}) \prod_{k=1}^{T} p\left(x_{t_k}|x_{t_0:t_{k-1}}\right) = p(x_{t_0}) \prod_{k=1}^{T} p\left(x_{t_k}|x_{t_{k-1}}\right).$$

Moreover, we note that

$$x_{t_k}|x_{t_{k-1}} \sim \mathcal{N}\left(F_{t_k}x_{t_{k-1}}, K_{t_k|t_{k-1}}\right)$$

with $F_{t_k} = K_{t_k,t_{k-1}} K^{-1}_{t_{k-1},t_{k-1}}$, and we recall that if X is a vector of $(p+1)$ i.i.d. standard normal, M a deterministic vector and L a square matrix both with appropriate dimensions, then $M + LX$ is a Gaussian vector with mean M and covariance matrix LL^T. It then follows that the full state space representation reads

$$\begin{cases} x_{t_0} \sim \mathcal{N}(0, K_{t_0,t_0}) \\ x_{t_k} = F_{t_k} x_{t_{k-1}} + L_{t_k|t_{k-1}} \xi_{t_k} \\ y_{t_k} = m(t_k) + H^T x_{t_k} + \epsilon_{t_k} \end{cases}$$

where $(\xi_t)_{t\geq 0}$ is a $(p+1)$-dimensional standard Gaussian white noise.

B.6 Proof of Proposition 3.11

Objective: In this section we prove that the pM-GP filter is equivalent to Gaussian process regression under a spectral Matérn kernel (Proposition 3.11).

Proof Following the notations of Definition 3.10 and Proposition 3.11, and using the result of Appendix B.5, we first note that by construction the processes $\{\dots, ({}^i_c x_t)_{t\geq 0}, ({}^i_s x_t)_{t\geq 0}, \dots\}$ are mutually independent and both $({}^i_c x_t)_{t\geq 0}$ and $({}^i_s x_t)_{t\geq 0}$ are pDGP with mean 0 and Matérn covariance function $k_{\text{ma}}(\tau; k_{0i}, l_i, p + \frac{1}{2})$. Writting

$${}^i_* x_t = \left({}^i_* z_t, {}^i_* z^{(1)}_t, \dots, {}^i_* z^{(p)}_t\right)$$

where $*$ is either c or s, it is easy to see that the observations process of the pM-GP filter can be written down as

$$y_t = z_t + \epsilon_t,$$

where

$$z_t = m(t) + \sum_{i=0}^{n} {}^i_c z_t \cos(\omega_i t) + {}^i_s z_t \sin(\omega_i t). \tag{B.2}$$

It is also easy to see that $(z_t)_{t\geq 0}$ is a Gaussian process with mean function m, and by mutual independence of $\{\dots, ({}^i_c x_t)_{t\geq 0}, ({}^i_s x_t)_{t\geq 0}, \dots\}$ we also have that

$$\begin{aligned} \operatorname{cov}(z_u, z_v) &= \sum_{i=0}^{n} \left(\operatorname{cov}({}^i_c z_u, {}^i_c z_v) \cos(\omega_i u) \cos(\omega_i v) + \operatorname{cov}({}^i_s z_u, {}^i_s z_v) \sin(\omega_i u) \sin(\omega_i v) \right) \\ &= \sum_{i=0}^{n} k_{\text{ma}}\left(\tau; k_{0i}, l_i, p + \frac{1}{2}\right) \cos\left(\omega_i (u - v)\right). \end{aligned}$$

This proves that $(z_t)_{t\geq 0}$ has the same law as $(\hat{z}_t)_{t\geq 0}$. Given that $(\epsilon_t)_{t\geq 0}$ has the same law as $(\hat{\epsilon}_t)_{t\geq 0}$, it follows that $(y_t)_{t\geq 0}$ has the same law as $(\hat{y}_t)_{t\geq 0}$, which concludes the proof. ■

B.7 Solution to the Forecasting Problem for the pM-GP Filter

Objective: In this section we derive the solution to the forecasting problem for the pM-GP filter (Equations (3.18, 3.19, 3.20)); in particular we provide an iterative algorithm to compute the posterior distribution over future values of the latent time series given historical noisy observations: $p(z_t|y_{t_k}\dots y_{t_0})$ for $0 < t_0 < \cdots < t_k < t$.

Derivation: The derivation is almost identical to the Bayesian derivation of the Kalman filter, except for the additional trend term m in the observation equation. We note from Equation (3.17) that the processes $(y_t)_{t\geq 0}$ and $(\boldsymbol{x}_t)_{t\geq 0}$ are jointly Gaussian. Hence, $\boldsymbol{x}_{t_0}$ and y_{t_0} are jointly Gaussian, which implies $\boldsymbol{x}_{t_0}|y_{t_0}$ is Gaussian too. Using Equation (3.17) and Proposition 3.11 we get:

$$\forall t > 0,\ \mathrm{E}(\boldsymbol{x}_t) = 0,\ \mathrm{E}(y_t) = m(t),\ \mathrm{E}(z_t) = m(t)$$

$$\begin{aligned}
\mathrm{cov}\left(\boldsymbol{x}_t, \boldsymbol{x}_t\right) &:= \boldsymbol{K}_{t,t},\\
\mathrm{var}(y_t) &= \boldsymbol{H}_t^T \boldsymbol{K}_{t,t}\boldsymbol{H}_t + \sigma^2,\\
\mathrm{cov}\left(y_t, z_t\right) = \mathrm{cov}\left(z_t, z_t\right) &= \boldsymbol{H}_t^T \boldsymbol{K}_{t,t}\boldsymbol{H}_t,
\end{aligned}$$

$$\begin{aligned}
\mathrm{cov}\left(\boldsymbol{x}_t, y_t\right) &:= \mathrm{E}\left(\boldsymbol{x}_t y_t\right) - \mathrm{E}\left(\boldsymbol{x}_t\right)\mathrm{E}\left(y_t\right)\\
&= \mathrm{E}\left(\boldsymbol{x}_t y_t\right)\\
&= \mathrm{E}\left(\boldsymbol{x}_t\right) m(t) + \mathrm{E}\left(\boldsymbol{x}_t \boldsymbol{H}_t^T \boldsymbol{x}_t\right) + \mathrm{E}\left(\boldsymbol{x}_t \epsilon_t\right)\\
&= \mathrm{E}\left(\boldsymbol{x}_t \boldsymbol{x}_t^T\right)\boldsymbol{H}_t\\
&= \boldsymbol{K}_{t,t}\boldsymbol{H}_t.
\end{aligned}$$

Moreover, using standard Gaussian identities, we get:

$$\mathrm{E}\left(\boldsymbol{x}_t|y_t\right) = \frac{1}{\boldsymbol{H}_t^T \boldsymbol{K}_{t,t}\boldsymbol{H}_t + \sigma^2}\boldsymbol{K}_{t,t}\boldsymbol{H}_t\left(y_t - m(t)\right)$$

$$\operatorname{cov}(\boldsymbol{x}_t|y_t) = \boldsymbol{K}_{t,t} - \frac{1}{\boldsymbol{H}_t^T \boldsymbol{K}_{t,t} \boldsymbol{H}_t + \sigma^2} \boldsymbol{K}_{t,t} \boldsymbol{H}_t \boldsymbol{H}_t^T \boldsymbol{K}_{t,t}^T$$

$$\begin{aligned} \mathrm{E}(z_t|y_t) &= m(t) + \frac{1}{\boldsymbol{H}_t^T \boldsymbol{K}_{t,t} \boldsymbol{H}_t + \sigma^2} \boldsymbol{H}_t^T \boldsymbol{K}_{t,t} \boldsymbol{H}_t (y_t - m(t)) \\ &= m(t) + \boldsymbol{H}_t^T \mathrm{E}(\boldsymbol{x}_t|y_t) \end{aligned}$$

$$\begin{aligned} \operatorname{cov}(z_t|y_t) &= \boldsymbol{H}_t^T \boldsymbol{K}_{t,t} \boldsymbol{H}_t - \frac{\boldsymbol{H}_t^T \boldsymbol{K}_{t,t} \boldsymbol{H}_t \boldsymbol{H}_t^T \boldsymbol{K}_{t,t}^T \boldsymbol{H}_t}{\boldsymbol{H}_t^T \boldsymbol{K}_{t,t} \boldsymbol{H}_t + \sigma^2} \\ &= \boldsymbol{H}_t^T \operatorname{cov}(\boldsymbol{x}_t|y_t) \boldsymbol{H}_t. \end{aligned}$$

Hence, with $\boldsymbol{m}_{t_0}^- = 0$, $\boldsymbol{P}_{t_0}^- = \boldsymbol{K}_{t_0,t_0}$ and

$$\forall t \begin{cases} v_t^- & := \boldsymbol{H}_t^T \boldsymbol{P}_t^- \boldsymbol{H}_t + \sigma^2 \\ \boldsymbol{e}_t^- & := y_t - m(t) - \boldsymbol{H}_t^T \boldsymbol{m}_t^- \\ \boldsymbol{G}_t & := \frac{1}{v_t^-} \boldsymbol{P}_t^- \boldsymbol{H}_t \\ \boldsymbol{m}_t & := \boldsymbol{m}_t^- + \boldsymbol{e}_t^- \boldsymbol{G}_t \\ \boldsymbol{P}_t & := \boldsymbol{P}_t^- - v_t^- \boldsymbol{G}_t \boldsymbol{G}_t^T \\ v_t & := \boldsymbol{H}_t^T \boldsymbol{P}_t \boldsymbol{H}_t \end{cases}, \tag{B.3}$$

we get $\boldsymbol{x}_{t_0}|y_{t_0} \sim \mathcal{N}(\boldsymbol{m}_{t_0}, \boldsymbol{P}_{t_0})$ and $z_{t_0}|y_{t_0} \sim \mathcal{N}(m(t) + \boldsymbol{H}_{t_0}^T \boldsymbol{m}_{t_0}, v_{t_0})$. In order to derive the remaining steps, we need the following lemma.

Lemma B.1 *Let X be a multivariate Gaussian with mean μ_X and covariance matrix Σ_X. If conditional on X, Y is a multivariate Gaussian with mean $MX + A$ and covariance matrix Σ_Y^c where M, A and Σ_Y^c do not depend on X, then (X, Y) is a jointly Gaussian vector with mean*

$$\mu_{X;Y} = \begin{bmatrix} \mu_X \\ M\mu_X + A \end{bmatrix}$$

and covariance matrix

$$\Sigma_{X;Y} = \begin{bmatrix} \Sigma_X & \Sigma_X M^T \\ M\Sigma_X & \Sigma_Y^c + M\Sigma_X M^T \end{bmatrix}.$$

Proof To prove this lemma we introduce two vectors $\tilde{X}$ and $\tilde{Y}$ whose lengths are the same as those of X and Y respectively, and such that $(\tilde{X}, \tilde{Y})$ is jointly Gaussian with mean $\mu_{X;Y}$ and covariance matrix $\Sigma_{X;Y}$. We then prove that the (marginal) distribution of $\tilde{X}$ is the same as the distribution of X and that the distribution of $\tilde{Y}|\tilde{X} = x$ is the same as $Y|X = x$ for any x, which is sufficient to conclude that (X, Y) and $(\tilde{X}, \tilde{Y})$ have the same distribution.

It is obvious from the joint $(\tilde{X}, \tilde{Y})$ that $\tilde{X}$ is Gaussian distribution with mean μ_X and covariance matrix Σ_X. As for the distribution of $\tilde{Y}$ conditional on $\tilde{X} = x$, it follows from the usual Gaussian identities that it is Gaussian with mean

$$M\mu_X + c + M\Sigma_X\Sigma_X^{-1}(x - \mu_X) = Mx + c,$$

and covariance matrix

$$\Sigma_Y^c + M\Sigma_X M^T - M\Sigma_X\Sigma_X^{-1}\Sigma_X^T M^T = \Sigma_Y^c,$$

which is the same distribution as that of $Y|X = x$ since the covariance matrix Σ_X is symmetric. This concludes our proof. ■

For $k > 0$ and $t > t_{k-1}$ we proceed by iteration. We assume that

$$\begin{cases} \boldsymbol{x}_{t_{k-1}}|y_{t_0:t_{k-1}} & \sim \mathcal{N}(\boldsymbol{m}_{t_{k-1}}, \boldsymbol{P}_{t_{k-1}}) \\ z_{t_{k-1}}|y_{t_0:t_{k-1}} & \sim \mathcal{N}(m(t_{k-1}) + \boldsymbol{H}_{t_{k-1}}^T \boldsymbol{m}_{t_{k-1}}, v_{t_{k-1}}) \end{cases} \tag{B.4}$$

using the definitions in Equations (B.3), and we would like to prove that

$$\begin{cases} \boldsymbol{x}_t|y_{t_0:t_{k-1}} & \sim \mathcal{N}(\boldsymbol{m}_t^-, \boldsymbol{P}_t^-) \\ y_t|y_{t_0:t_{k-1}} & \sim \mathcal{N}(m(t) + \boldsymbol{H}_t^T \boldsymbol{m}_t^-, v_t^-) \\ z_t|y_{t_0:t_{k-1}} & \sim \mathcal{N}(m(t) + \boldsymbol{H}_t^T \boldsymbol{m}_t^-, v_t^- - \sigma^2), \end{cases} \tag{B.5}$$

with

$$\begin{cases} \boldsymbol{m}_t^- & = \boldsymbol{F}_t \boldsymbol{m}_{t_{k-1}} \\ \boldsymbol{P}_t^- & = \boldsymbol{F}_t \boldsymbol{P}_{t_{k-1}} \boldsymbol{F}_t^T + \boldsymbol{K}_{t|t_{k-1}} \end{cases}.$$

To do so, we note that

$$p\left(\boldsymbol{x}_t, \boldsymbol{x}_{t_{k-1}}|y_{t_0:t_{k-1}}\right) = p\left(\boldsymbol{x}_t|\boldsymbol{x}_{t_{k-1}}, y_{t_0:t_{k-1}}\right) p\left(\boldsymbol{x}_{t_{k-1}}|y_{t_0:t_{k-1}}\right)$$

$$= p\left(\boldsymbol{x}_t|\boldsymbol{x}_{t_{k-1}}\right) p\left(\boldsymbol{x}_{t_{k-1}}|y_{t_0:t_{k-1}}\right)$$

where the first equality results from Bayes' rule and the second result from the Markov property of $(\boldsymbol{x}_t)_{t\geq 0}$. As

$$\begin{aligned}\boldsymbol{x}_t|\boldsymbol{x}_{t_{k-1}}, y_{t_0:t_{k-1}} &\sim \mathcal{N}\left(\boldsymbol{F}_t\boldsymbol{x}_{t_{k-1}}, \boldsymbol{K}_{t|t_{k-1}}\right)\\ \boldsymbol{x}_{t_{k-1}}|y_{t_0:t_{k-1}} &\sim \mathcal{N}(\boldsymbol{m}_{t_{k-1}}, \boldsymbol{P}_{t_{k-1}}),\end{aligned}$$

it follows from Lemma B.1 that

$$\begin{aligned}\boldsymbol{x}_t|y_{t_0:t_{k-1}} &\sim \mathcal{N}(\boldsymbol{F}_t\boldsymbol{m}_{t_{k-1}}, \boldsymbol{K}_{t|t_{k-1}} + \boldsymbol{F}_t\boldsymbol{P}_{t_{k-1}}\boldsymbol{F}_t^T)\\ &\sim \mathcal{N}(\boldsymbol{m}_t^-, \boldsymbol{P}_t^-).\end{aligned}$$

Moreover, as $(y_t)_{t\geq 0}$ is a Gaussian process, $y_t|y_{t_0:t_{k-1}}$ is Gaussian. What's more, as $y_t = m(t) + \boldsymbol{H}_t^T\boldsymbol{x}_t + \epsilon_t$ and $\epsilon_t \perp y_{t_0:t_{k-1}}$, we have that:

$$\begin{aligned}\mathrm{E}(y_t|y_{t_0:t_{k-1}}) &= m(t) + \boldsymbol{H}_t^T\mathrm{E}\left(\boldsymbol{x}_t|y_{t_0:t_{k-1}}\right)\\ &= m(t) + \boldsymbol{H}_t^T\boldsymbol{m}_t^-\end{aligned}$$

and

$$\begin{aligned}\mathrm{var}(y_t|y_{t_0:t_{k-1}}) &= \boldsymbol{H}_t^T\mathrm{cov}\left(\boldsymbol{x}_t, \boldsymbol{x}_t|y_{t_0:t_{k-1}}\right)\boldsymbol{H}_t + \mathrm{E}(\epsilon_t^2)\\ &= \boldsymbol{H}_t^T\boldsymbol{P}_t^-\boldsymbol{H}_t + \sigma^2\\ &= v_t^-.\end{aligned}$$

The distribution $z_t|y_{t_0:t_{k-1}}$ is obtained in a similar fashion. More generally, as the processes $(\boldsymbol{x}_t)_{t\geq 0}$ and $(y_t)_{t\geq 0}$ are jointly Gaussian, the random variables $\boldsymbol{x}_t$ and y_t are also jointly Gaussian conditional on $y_{t_0:t_{k-1}}$ and

$$\begin{aligned}\mathrm{cov}(\boldsymbol{x}_t, y_t|y_{t_0:t_{k-1}}) &= \mathrm{cov}\left(\boldsymbol{x}_t, m(t) + \boldsymbol{H}_t^T\boldsymbol{x}_t + \epsilon_t|y_{t_0:t_{k-1}}\right)\\ &= \mathrm{cov}\left(\boldsymbol{x}_t, \boldsymbol{H}_t^T\boldsymbol{x}_t + \epsilon_t|y_{t_0:t_{k-1}}\right)\\ &= \mathrm{cov}\left(\boldsymbol{x}_t, \boldsymbol{H}_t^T\boldsymbol{x}_t|y_{t_0:t_{k-1}}\right) + \mathrm{cov}\left(\boldsymbol{x}_t, \epsilon_t|y_{t_0:t_{k-1}}\right)\\ &= \mathrm{cov}\left(\boldsymbol{x}_t, \boldsymbol{x}_t|y_{t_0:t_{k-1}}\right)\boldsymbol{H}_t + \mathrm{E}(\boldsymbol{x}_t\epsilon_t|y_{t_0:t_{k-1}})\\ &= \boldsymbol{P}_t^-\boldsymbol{H}_t + \mathrm{E}(\boldsymbol{x}_t|y_{t_0:t_{k-1}})\mathrm{E}(\epsilon_t)\\ &= \boldsymbol{P}_t^-\boldsymbol{H}_t.\end{aligned}$$

Finally, under the assumptions of Equations (B.4), which we recall imply Equations (B.5), we would like to prove that

$$\begin{cases} \boldsymbol{x}_t|y_{t_0:t} & \sim \mathcal{N}(\boldsymbol{m}_t, \boldsymbol{P}_t) \\ z_t|y_{t_0:t} & \sim \mathcal{N}(m(t) + \boldsymbol{H}_t^T \boldsymbol{m}_t, v_t) \end{cases}. \tag{B.6}$$

We have previously established that $\boldsymbol{x}_t, y_t|y_{t_0:t_{k-1}}$ is Gaussian and we have derived the corresponding mean and covariance matrix. Noting that by definition

$$\boldsymbol{x}_t|(y_{t_0:t_{k-1}}, y_t) := \boldsymbol{x}_t|y_{t_0:t},$$

it follows from standard Gaussian identities that

$$\begin{aligned} \mathrm{E}\left(\boldsymbol{x}_t|y_{t_0:t}\right) &= \mathrm{E}\left(\boldsymbol{x}_t|y_{t_0:t_{k-1}}\right) + \frac{y_t - \mathrm{E}\left(y_t|y_{t_0:t_{k-1}}\right)}{\mathrm{var}\left(y_t|y_{t_0:t_{k-1}}\right)} \mathrm{cov}\left(\boldsymbol{x}_t, y_t|y_{t_0:t_{k-1}}\right) \\ &= \boldsymbol{m}_t^- + \frac{1}{v_t^-} \boldsymbol{P}_t^- \boldsymbol{H}_t \left(y_t - m(t) - \boldsymbol{H}_t^T \boldsymbol{m}_t^-\right) \\ &= \boldsymbol{m}_t^- + \boldsymbol{G}_t \boldsymbol{e}_t^- \\ &= \boldsymbol{m}_t \end{aligned}$$

and

$$\begin{aligned} \mathrm{cov}\left(\boldsymbol{x}_t, \boldsymbol{x}_t|y_{t_0:t}\right) &= \mathrm{cov}\left(\boldsymbol{x}_t, \boldsymbol{x}_t|y_{t_0:t_{k-1}}\right) - \frac{\mathrm{cov}\left(\boldsymbol{x}_t, y_t|y_{t_0:t_{k-1}}\right) \mathrm{cov}\left(\boldsymbol{x}_t, y_t|y_{t_0:t_{k-1}}\right)^T}{\mathrm{var}\left(y_t|y_{t_0:t_{k-1}}\right)} \\ &= \boldsymbol{P}_t^- - \frac{1}{v_t^-} \boldsymbol{P}_t^- \boldsymbol{H}_t \boldsymbol{H}_t^T \boldsymbol{P}_t^{-T} \\ &= \boldsymbol{P}_t^- - v_t^- \boldsymbol{G}_t \boldsymbol{G}_t^T \\ &= \boldsymbol{P}_t. \end{aligned} \tag{B.7}$$

This proves the first part of Equations (B.6). As for the second part, it is a direct consequence of

$$z_t = m(t) + \boldsymbol{H}_t^T \boldsymbol{x}_t.$$

B.8 Solution to the Constrained Optimisation Problem (3.22)

Objective: In this section we derive the solution to the constrained optimization problem (3.22).

Proof We start by recalling the problem of interest:

$$\begin{cases} \boldsymbol{\theta}_{t_k}, \xi_{t_k} = \underset{\boldsymbol{\theta},\xi}{\operatorname{argmin}} \; ||\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}}||^2 + c_k \xi^2 \\ \text{s.t.} \;\; \max\left(-\epsilon - \hat{\mathcal{L}}^l_{t_k}(\boldsymbol{\theta}), 0\right) \leq \xi \end{cases},$$

with $\hat{\mathcal{L}}^l_{t_k}(\boldsymbol{\theta}) = \mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}) + \nabla \mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})^T(\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}})$, $c_k > 0$ and $\epsilon \geq 0$. It is easy to note that when $\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}) > -\epsilon$, the solution to the optimization problem is $(\boldsymbol{\theta}_{t_{k-1}}, 0)$, so that we may focus on the case $\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}) \leq -\epsilon$. For $\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}) \leq -\epsilon$, the problem can then be rewritten as the convex optimization problem:

$$\begin{cases} \boldsymbol{\theta}_{t_k} = \underset{\boldsymbol{\theta},\xi}{\operatorname{argmin}} \; ||\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}}||^2 + c_k \xi^2 \\ \text{s.t.} \; -\epsilon - \mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}) - \nabla \mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})^T(\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}}) - \xi \leq 0. \end{cases}$$

As the constraints are linear and the domain of the objective is not restricted, Slater's condition is met and strong duality holds (see Boyd and Vandenberghe, 2004, §5.2.3). The minimizer is therefore obtained by setting the gradient of the Lagrangian

$$||\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}}||^2 + c_k \xi^2 + \lambda\left(-\epsilon - \mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}) - \xi - \nabla \mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})^T(\boldsymbol{\theta} - \boldsymbol{\theta}_{t_{k-1}})\right), \lambda \geq 0 \tag{B.8}$$

to 0. Setting the gradient with respect to $\boldsymbol{\theta}$ to 0 we get

$$\boldsymbol{\theta}_{t_k} = \boldsymbol{\theta}_{t_{k-1}} + \frac{\lambda^*}{2} \nabla \mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}). \tag{B.9}$$

Setting the derivative with respect to ξ to zero, we get

$$\xi^* = \frac{\lambda^*}{2c_k}. \tag{B.10}$$

Finally, plugging Equations (B.9) and (B.10) into Equation (B.8), we can rewrite the Lagrangian as

$$-\left(\frac{\|\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})\|^2}{4}+\frac{1}{4c}\right)(\lambda^*)^2-\left(\epsilon+\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})\right)\lambda^*,$$

which reaches its maximum at

$$\lambda^*=-2c_k\frac{\epsilon+\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})}{1+c_k\|\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})\|^2}. \tag{B.11}$$

Using Equations (B.9) and (B.11), together with the result established for the case $\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})>-\epsilon$, we conclude that

$$\boldsymbol{\theta}_{t_k}=\boldsymbol{\theta}_{t_{k-1}}+c_k\frac{\max\left(-\epsilon-\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}),0\right)}{1+c_k\|\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}})\|^2}\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta}_{t_{k-1}}),$$

which ends the proof. ■

B.9 Derivation of $\nabla\mathcal{L}^l_{t_k}$

Objective: In this section we derive $\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta})$. We assume that the trend function m is parametric and has parameters $\boldsymbol{\beta}$.

Derivation: We recall that $\mathcal{L}^l_{t_k}(\boldsymbol{\theta})$ is the logarithm of the probability density function of a Gaussian. To ease derivations we denote

$$\bar{m}_{t_k}(\boldsymbol{\theta}):=m(t_k)+\boldsymbol{H}^T_{t_k}\boldsymbol{m}^-_{t_k}\text{ and }\bar{v}_{t_k}(\boldsymbol{\theta}):=v^-_{t_k}$$

the mean and variance of the corresponding Gaussian, so that

$$\mathcal{L}^l_{t_k}(\boldsymbol{\theta})=-\frac{\log(2\pi)}{2}-\frac{\log\left(\bar{v}_{t_k}(\boldsymbol{\theta})\right)}{2}-\frac{\left(y_{t_k}-\bar{m}_{t_k}(\boldsymbol{\theta})\right)^2}{2\bar{v}_{t_k}(\boldsymbol{\theta})}. \tag{B.12}$$

It then follows that:

$$\nabla\mathcal{L}^l_{t_k}(\boldsymbol{\theta})=\left(-\frac{1}{2\bar{v}_{t_k}(\boldsymbol{\theta})}+\frac{\left(y_{t_k}-\bar{m}_{t_k}(\boldsymbol{\theta})\right)^2}{2\bar{v}_{t_k}(\boldsymbol{\theta})^2}\right)\nabla\bar{v}_{t_k}(\boldsymbol{\theta})+\left(\frac{y_{t_k}-\bar{m}_{t_k}(\boldsymbol{\theta})}{\bar{v}_{t_k}(\boldsymbol{\theta})}\right)\nabla\bar{m}_{t_k}(\boldsymbol{\theta}), \tag{B.13}$$

so that all we need to do is derive the gradients of $\bar{m}_{t_k}$ and $\bar{v}_{t_k}$ (using Equations (3.19) and (3.20)). To do so, we recall that $\boldsymbol{\theta}$ is made of the parameters of m that we denote $\boldsymbol{\beta}$, $\log \sigma$ and $\{\log k_{0i}, \log l_i, \log \omega_i\}_{i=0}^n$.

Derivatives with respect to $\boldsymbol{\beta}$:

$$\frac{\partial \bar{m}_{t_k}(\boldsymbol{\theta})}{\partial \boldsymbol{\beta}} = \frac{\partial m}{\partial \boldsymbol{\beta}}, \ \frac{\partial \bar{v}_{t_k}(\boldsymbol{\theta})}{\partial \boldsymbol{\beta}} = 0.$$

Derivatives with respect to $\log \sigma$:

$$\frac{\partial \bar{m}_{t_k}(\boldsymbol{\theta})}{\partial \log \sigma} = 0, \ \frac{\partial \bar{v}_{t_k}(\boldsymbol{\theta})}{\partial \log \sigma} = \frac{\partial \bar{v}_{t_k}(\boldsymbol{\theta})}{\partial \sigma^2} \frac{\mathrm{d}\sigma^2}{\mathrm{d} \log \sigma} = 2\sigma^2.$$

Derivatives with respect to $\log \omega_i$:

$$\begin{aligned}
\frac{\partial \bar{m}_{t_k}(\boldsymbol{\theta})}{\partial \log \omega_i} &= \frac{\partial \boldsymbol{H}_{t_k}^T}{\partial \log \omega_i} \boldsymbol{m}_{t_k}^- \\
\frac{\partial \bar{v}_{t_k}(\boldsymbol{\theta})}{\partial \log \omega_i} &= \frac{\partial \boldsymbol{H}_{t_k}^T}{\partial \log \omega_i} \boldsymbol{P}_{t_k}^- \boldsymbol{H}_{t_k} + \boldsymbol{H}_{t_k}^T \boldsymbol{P}_{t_k}^- \frac{\partial \boldsymbol{H}_{t_k}}{\partial \log \omega_i}
\end{aligned}$$

where $\frac{\partial \boldsymbol{H}_{t_k}^T}{\partial \log \omega_i}$ is identical to $\boldsymbol{H}_{t_k}^T$ except that all terms in ω_j, $j \neq i$ are set to 0, the term in $\cos(\omega_i t_k)$ becomes $-t_k \omega_i \sin(\omega_i t_k)$, and the term in $\sin(\omega_i t_k)$ becomes $t_k \omega_i \cos(\omega_i t_k)$.

Derivatives with respect to $\log k_{0i}$ and $\log l_i$: We recall that

$$\begin{aligned}
\bar{m}_{t_k}(\boldsymbol{\theta}) &= m(t_k) + \boldsymbol{H}_{t_k}^T \boldsymbol{F}_{t_k} \boldsymbol{m}_{t_{k-1}} \\
\bar{v}_{t_k}(\boldsymbol{\theta}) &= \boldsymbol{H}_{t_k}^T \boldsymbol{P}_{t_k}^- \boldsymbol{H}_{t_k} + \sigma^2
\end{aligned}$$

with $\boldsymbol{P}_{t_k}^- = \boldsymbol{F}_{t_k} \boldsymbol{P}_{t_{k-1}} \boldsymbol{F}_{t_k}^T + \boldsymbol{K}_{t_k|t_{k-1}}$. Hence, the only terms that depend on $\log k_{0i}$ and $\log l_i$ are $\boldsymbol{F}_{t_k}$ and $\boldsymbol{K}_{t_k|t_{k-1}}$, and partial derivatives of $\bar{m}_{t_k}$ and $\bar{v}_{t_k}$ with respect to $\log k_{0i}$ and $\log l_i$ are easily derived from that of $\boldsymbol{F}_{t_k}$ and $\boldsymbol{K}_{t_k|t_{k-1}}$:

$$\begin{aligned}
\frac{\partial \bar{m}_{t_k}(\boldsymbol{\theta})}{\partial \log k_{0i}} &= \boldsymbol{H}_{t_k}^T \frac{\partial \boldsymbol{F}_{t_k}}{\partial \log k_{0i}} \boldsymbol{m}_{t_{k-1}} \\
\frac{\partial \bar{m}_{t_k}(\boldsymbol{\theta})}{\partial \log l_i} &= \boldsymbol{H}_{t_k}^T \frac{\partial \boldsymbol{F}_{t_k}}{\partial \log l_i} \boldsymbol{m}_{t_{k-1}} \\
\frac{\partial \bar{v}_{t_k}(\boldsymbol{\theta})}{\partial \log k_{0i}} &= \boldsymbol{H}_{t_k}^T \frac{\partial \boldsymbol{P}_{t_k}^-}{\partial \log k_{0i}} \boldsymbol{H}_{t_k} \\
\frac{\partial \bar{v}_{t_k}(\boldsymbol{\theta})}{\partial \log l_i} &= \boldsymbol{H}_{t_k}^T \frac{\partial \boldsymbol{P}_{t_k}^-}{\partial \log l_i} \boldsymbol{H}_{t_k}
\end{aligned}$$

with

$$\frac{\partial \boldsymbol{P}_{t_k}^-}{\partial \log k_{0i}} = \frac{\partial \boldsymbol{K}_{t_k|t_{k-1}}}{\partial \log k_{0i}} + \frac{\partial \boldsymbol{F}_{t_k}}{\partial \log k_{0i}} \boldsymbol{P}_{t_{k-1}} \boldsymbol{F}_{t_k}^T + \boldsymbol{F}_{t_k} \boldsymbol{P}_{t_{k-1}} \frac{\partial \boldsymbol{F}_{t_k}^T}{\partial \log k_{0i}}$$
$$\frac{\partial \boldsymbol{P}_{t_k}^-}{\partial \log l_i} = \frac{\partial \boldsymbol{K}_{t_k|t_{k-1}}}{\partial \log l_i} + \frac{\partial \boldsymbol{F}_{t_k}}{\partial \log l_i} \boldsymbol{P}_{t_{k-1}} \boldsymbol{F}_{t_k}^T + \boldsymbol{F}_{t_k} \boldsymbol{P}_{t_{k-1}} \frac{\partial \boldsymbol{F}_{t_k}^T}{\partial \log l_i}.$$

Appendix C

Derivations of Chapters 4 and 5

We begin by recalling Kolmogorov's extension theorem, which we will use to prove the existence of *derivative Gaussian processes* and *string Gaussian processes*.

Theorem C.1 *(Kolmogorov's extension theorem, (Øksendal, 2003, Theorem 2.1.5)) Let I be an interval, let all $t_1, \dots, t_i \in I$, $i, n \in \mathbb{N}^*$, let $\nu_{t_1,\dots,t_i}$ be probability measures on $\mathbb{R}^{ni}$ such that:*

$$\nu_{t_{\pi(1)},\dots,t_{\pi(i)}}(F_{\pi(1)}, \dots, F_{\pi(i)}) = \nu_{t_1,\dots,t_i}(F_1, \dots, F_i) \tag{C.1}$$

for all permutations π on $\{1, \dots, i\}$ and

$$\nu_{t_1,\dots,t_i}(F_1, \dots, F_i) = \nu_{t_1,\dots,t_i,t_{i+1},\dots,t_{i+m}}(F_1, \dots, F_i, \mathbb{R}^n, \dots, \mathbb{R}^n) \tag{C.2}$$

for all $m \in \mathbb{N}^*$ where the set on the right hand side has a total of $i + m$ factors. Then there exists a probability space $(\Omega, \mathcal{F}, \mathbb{P})$ and an $\mathbb{R}^n$ valued stochastic process $(X_t)_{t\in I}$ on Ω,

$$X_t : \Omega \to \mathbb{R}^n$$

such that

$$\nu_{t_1,\dots,t_i}(F_1, \dots, F_i) = \mathbb{P}(X_{t_1} \in F_1, \dots, X_{t_i} \in F_i) \tag{C.3}$$

for all $t_1, \dots, t_i \in I$, $i \in \mathbb{N}^*$ and for all Borel sets $F_1, \dots, F_i$.

It is easy to see that every stochastic process satisfies the permutation and marginalisation conditions (C.1) and (C.2). The power of Kolmogorov's extension theorem is that it states that those two conditions are sufficient to guarantee the existence of a stochastic process.

C.1 Proof of Proposition 4.1

In this section we prove Proposition 4.1, which we recall below.

Proposition 4.1 (Derivative Gaussian processes)
Let I be an interval, $k : I \times I \to \mathbb{R}$ a $\mathcal{C}^2$ symmetric positive semi-definite function[1], $m : I \to \mathbb{R}$ a $\mathcal{C}^1$ function.
(A) There exists a $\mathbb{R}^2$-valued stochastic process $(D_t)_{t\in I}$, $D_t = (z_t, z'_t)$, such that for all $t_1, \ldots, t_n \in I$,

$$(z_{t_1}, \ldots, z_{t_n}, z'_{t_1}, \ldots, z'_{t_n})$$

is a Gaussian vector with mean

$$\left(m(t_1), \ldots, m(t_n), \frac{\mathrm{d}m}{\mathrm{dt}}(t_1), \ldots, \frac{\mathrm{d}m}{\mathrm{dt}}(t_n)\right)$$

and covariance matrix such that

$$\mathrm{cov}(z_{t_i}, z_{t_j}) = k(t_i, t_j), \quad \mathrm{cov}(z_{t_i}, z'_{t_j}) = \frac{\partial k}{\partial y}(t_i, t_j), \quad \text{and} \quad \mathrm{cov}(z'_{t_i}, z'_{t_j}) = \frac{\partial^2 k}{\partial x \partial y}(t_i, t_j).$$

We herein refer to $(D_t)_{t\in I}$ as a **derivative Gaussian process**.
(B) $(z_t)_{t\in I}$ is a Gaussian process with mean function m, covariance function k and that is $\mathcal{C}^1$ in the L^2 (mean square) sense.
(C) $(z'_t)_{t\in I}$ is a Gaussian process with mean function $\frac{\mathrm{d}m}{\mathrm{dt}}$ and covariance function $\frac{\partial^2 k}{\partial x \partial y}$. Moreover, $(z'_t)_{t\in I}$ is the L^2 derivative of the process $(z_t)_{t\in I}$.
Proof

C.1.1 Proof of Proposition 4.1 (A)

Firstly, we need to show that the matrix suggested in the proposition as the covariance matrix of $(z_{t_1}, \ldots, z_{t_n}, z'_{t_1}, \ldots, z'_{t_n})$ is indeed positive semi-definite. To do so, we will show that it is the limit of positive definite matrices (which is sufficient to conclude it is positive semi-definite, as $x^T M_n x \geq 0$ for a convergent sequence of positive definite matrices implies $x^T M_\infty x \geq 0$).

[1] $\mathcal{C}^1$ (resp. $\mathcal{C}^2$) functions denote functions that are once (resp. twice) continuously differentiable on their domains.

Let k be as in the proposition, h such that $\forall i \leq n,\ t_i + h \in I$ and $(\tilde{z}_t)_{t\in I}$ be a Gaussian process with covariance function k. The vector

$$\left(\tilde{z}_{t_1}, \ldots, \tilde{z}_{t_n}, \frac{\tilde{z}_{t_1+h} - \tilde{z}_{t_1}}{h}, \ldots, \frac{\tilde{z}_{t_n+h} - \tilde{z}_{t_n}}{h}\right)$$

is a Gaussian vector whose covariance matrix is positive definite and such that

$$\operatorname{cov}\left(\tilde{z}_{t_i}, \tilde{z}_{t_j}\right) = k(t_i, t_j), \tag{C.4}$$

$$\operatorname{cov}\left(\tilde{z}_{t_i}, \frac{\tilde{z}_{t_j+h} - \tilde{z}_{t_j}}{h}\right) = \frac{k(t_i, t_j + h) - k(t_i, t_j)}{h}, \tag{C.5}$$

and

$$\begin{aligned}&\operatorname{cov}\left(\frac{\tilde{z}_{t_i+h} - \tilde{z}_{t_i}}{h}, \frac{\tilde{z}_{t_j+h} - \tilde{z}_{t_j}}{h}\right)\\ &= \frac{1}{h^2}\left(k(t_i + h, t_j + h) - k(t_i + h, t_j) - k(t_i, t_j + h) + k(t_i, t_j)\right).\end{aligned} \tag{C.6}$$

As k is $\mathcal{C}^2$, $h \to k(x, y + h)$ admits a second order Taylor expansion about $h = 0$ for every x, and we have:

$$k(x, y + h) = k(x, y) + \frac{\partial k}{\partial y}(x, y)h + \frac{1}{2}\frac{\partial^2 k}{\partial y^2}(x, y)h^2 + o(h^2) = k(y + h, x). \tag{C.7}$$

Similarly, $h \to k(x + h, y + h)$ admits a second order Taylor expansion about $h = 0$ for every $x,\ y$ and we have:

$$\begin{aligned}k(x + h, y + h) = k(x, y) + \left[\frac{\partial k}{\partial x}(x, y) + \frac{\partial k}{\partial y}(x, y)\right] h + \left[\frac{\partial^2 k}{\partial x \partial y}(x, y) + \frac{1}{2}\frac{\partial^2 k}{\partial x^2}(x, y)\right.\\ \left. + \frac{1}{2}\frac{\partial^2 k}{\partial y^2}(x, y)\right] h^2 + o(h^2).\end{aligned} \tag{C.8}$$

Hence,

$$k(t_i, t_j + h) - k(t_i, t_j) = \frac{\partial k}{\partial y}(t_i, t_j)h + o(h), \tag{C.9}$$

and

$$k(t_i+h,t_j+h)-k(t_i+h,t_j)-k(t_i,t_j+h)+k(t_i,t_j)=\frac{\partial^2 k}{\partial x\partial y}(t_i,t_j)h^2+o(h^2). \tag{C.10}$$

Dividing Equation (C.9) by h, dividing Equation (C.10) by h^2, and taking the limits, we obtain:

$$\lim_{h\to 0}\ \mathrm{cov}\left(\tilde{z}_{t_i},\frac{\tilde{z}_{t_j+h}-\tilde{z}_{t_j}}{h}\right)=\frac{\partial k}{\partial y}(t_i,t_j),$$

and

$$\lim_{h\to 0}\ \mathrm{cov}\left(\frac{\tilde{z}_{t_i+h}-\tilde{z}_{t_i}}{h},\frac{\tilde{z}_{t_j+h}-\tilde{z}_{t_j}}{h}\right)=\frac{\partial^2 k}{\partial x\partial y}(t_i,t_j),$$

which corresponds to the covariance structure of Proposition 4.1. In other words the proposed covariance structure is indeed positive semi-definite.

Let $\nu^{\mathcal{N}}_{t_1,\dots,t_n}$ be the Gaussian probability measure corresponding to the joint distribution of $(z_{t_1},\dots,z_{t_n},z'_{t_1},\dots,z'_{t_n})$ as per the Proposition 4.1, and let $\nu^{D}_{t_1,\dots,t_n}$ be the measure on the Borel σ-algebra $\mathcal{B}(\underbrace{\mathbb{R}^2\times\cdots\times\mathbb{R}^2}_{n\text{ times}})$ such that for any $2n$ intervals $I_{11},I_{12},\dots,I_{n1},I_{n2}$,

$$\nu^{D}_{t_1,\dots,t_n}(I_{11}\times I_{12},\dots,I_{n1}\times I_{n2}):=\nu^{\mathcal{N}}_{t_1,\dots,t_n}(I_{11},\dots,I_{n1},I_{12},\dots,I_{n2}). \tag{C.11}$$

The measures $\nu^{D}_{t_1,\dots,t_n}$ are the finite dimensional measures corresponding to the stochastic object $(D_t)_{t\in I}$ sampled at times $t_1,\dots,t_n$. They satisfy the time permutation and marginalisation conditions of Kolmogorov's extension theorem as the Gaussian measures $\nu^{\mathcal{N}}_{t_1,\dots,t_n}$ do. Hence, the $\mathbb{R}^2$-valued stochastic process $(D_t)_{t\in I}$ defined in Proposition 4.1 does exist.

C.1.2 Proof of Proposition 4.1 (B)

That $(z_t)_{t\in I}$ is a Gaussian process results from the fact that the marginals $(z_{t_1},\dots,z_{t_n})$ are Gaussian vectors with mean $(m(t_1),\dots,m(t_n))$ and covariance matrix $[k(t_i,t_j)]_{i,j\in[1..n]}$. The fact that $(z_t)_{t\in I}$ is $\mathcal{C}^1$ in the L^2 sense is a direct consequence of the twice continuous differentiability of k.

C.1.3 Proof of Proposition 4.1 (C)

In effect, it follows from Proposition 4.1(A) that $\frac{z_{t+h}-z_t}{h} - z'_t$ is a Gaussian random variable with mean

$$\frac{m(t+h)-m(t)}{h} - \frac{\mathrm{d}m}{\mathrm{d}t}(t)$$

and variance

$$\frac{k(t+h,t+h) - 2k(t+h,t) + k(t,t) - 2\frac{\partial k}{\partial y}(t+h,t)h + 2\frac{\partial k}{\partial y}(t,t)h + \frac{\partial^2 k}{\partial x \partial y}(t,t)h^2}{h^2}.$$

Taking the second order Taylor expansion of the numerator in the fraction above about $h = 0$ we get $o(h^2)$, hence

$$\lim_{h\to 0} \mathrm{Var}\left(\frac{z_{t+h}-z_t}{h} - z'_t\right) = 0.$$

We also have

$$\lim_{h\to 0} \mathrm{E}\left(\frac{z_{t+h}-z_t}{h} - z'_t\right) = \frac{\mathrm{d}m}{dt}(t) - \mathrm{E}(z'_t) = 0.$$

Therefore,

$$\lim_{h\to 0} \mathrm{E}\left[\left(\frac{z_{t+h}-z_t}{h} - z'_t\right)^2\right] = 0,$$

which proves that (z'_t) is the L^2 derivative of (z_t). The fact that (z'_t) is a Gaussian process with mean function $\frac{\mathrm{d}m}{dt}$ and covariance function $\frac{\partial^2 k}{\partial x \partial y}$ is a direct consequence of the distribution of the marginals $(z'_{t_1}, \ldots, z'_{t_n})$. Moreover, the continuity of (z'_t) in the L^2 sense is a direct consequence of the continuity of $\frac{\partial^2 k}{\partial x \partial y}$ (see Rasmussen and Williams, 2005, p. 81 4.1.1). ■

C.2 Proof of Theorem 4.2

In this section we prove Theorem 4.2 which we recall below.

Theorem 4.2 (String Gaussian process)
Let $a_0 < \cdots < a_k < \cdots < a_K$, $I = [a_0, a_K]$ and let $\mathcal{N}(x|\mu, \Sigma)$ be the multivariate Gaussian density with mean vector μ and covariance matrix Σ. Furthermore, let $(m_k : [a_{k-1}, a_k] \to \mathbb{R})_{k\in[1..K]}$ be $\mathcal{C}^1$ functions, and $(k_k : [a_{k-1}, a_k] \times [a_{k-1}, a_k] \to \mathbb{R})_{k\in[1..K]}$ be $\mathcal{C}^3$ symmetric positive semi-definite functions, neither degenerate at a_{k-1}, nor degenerate at a_k given a_{k-1}.

(A) There exists an $\mathbb{R}^2$-valued stochastic process $(SD_t)_{t\in I}$, $SD_t = (z_t, z'_t)$ satisfying the following conditions:
1) The probability density of $(SD_{a_0}, \ldots, SD_{a_K})$ reads:

$$p_b(x_0, \ldots, x_K) := \prod_{k=0}^{K} \mathcal{N}\left(x_k | \mu_k^b, \Sigma_k^b\right)$$

$$\text{where:} \quad \Sigma_0^b = {}_1\mathbf{K}_{a_0;a_0}, \quad \forall\ k > 0 \quad \Sigma_k^b = {}_k\mathbf{K}_{a_k;a_k} - {}_k\mathbf{K}_{a_k;a_{k-1}}\ {}_k\mathbf{K}^{-1}_{a_{k-1};a_{k-1}}\ {}_k\mathbf{K}^{T}_{a_k;a_{k-1}},$$

$$\mu_0^b = {}_1\mathbf{M}_{a_0}, \quad \forall\ k > 0 \quad \mu_k^b = {}_k\mathbf{M}_{a_k} + {}_k\mathbf{K}_{a_k;a_{k-1}}\ {}_k\mathbf{K}^{-1}_{a_{k-1};a_{k-1}}(x_{k-1} - {}_k\mathbf{M}_{a_{k-1}}),$$

$$\text{with} \quad {}_k\mathbf{K}_{u;v} = \begin{bmatrix} k_k(u,v) & \frac{\partial k_k}{\partial y}(u,v) \\ \frac{\partial k_k}{\partial x}(u,v) & \frac{\partial^2 k_k}{\partial x \partial y}(u,v) \end{bmatrix}, \quad \text{and} \quad {}_k\mathbf{M}_u = \begin{bmatrix} m_k(u) \\ \frac{dm_k}{dt}(u) \end{bmatrix}.$$

2) Conditional on $(SD_{a_k} = x_k)_{k\in[0..K]}$, the restrictions $(SD_t)_{t\in]a_{k-1},a_k[}$, $k \in [1..K]$ are **independent conditional derivative Gaussian processes**, respectively with unconditional mean function m_k and unconditional covariance function k_k and that are conditioned to take values x_{k-1} and x_k at a_{k-1} and a_k respectively. We refer to $(SD_t)_{t\in I}$ as a **string derivative Gaussian process**, and to its first coordinate $(z_t)_{t\in I}$ as a **string Gaussian process** namely,

$$(z_t)_{t\in I} \sim \mathcal{SGP}(\{a_k\}, \{m_k\}, \{k_k\}).$$

(B) The **string Gaussian process** $(z_t)_{t\in I}$ defined in (A) is $\mathcal{C}^1$ in the L^2 sense and its L^2 derivative is the process $(z'_t)_{t\in I}$ defined in (A).
Proof

C.2.1 Proof of Theorem 4.2 (A)

We will once again turn to Kolmogorov's extension theorem to prove the existence of the stochastic process $(SD_t)_{t\in I}$. The core of the proof is in the finite dimensional measures implied by Theorem 4.2 (A-1) and (A-2). Let $\left\{t_i^k \in]a_{k-1}, a_k[\right\}_{i\in[1..N_k],k\in[1..K]}$ be n times. We first formally construct the finite dimensional measures implied by Theorem 4.2 (A-1) and (A-2), and then verify that they satisfy the conditions of Kolmogorov's extension theorem.

Let us define the measure $\nu^{SD}_{t^1_1,\dots,t^1_{N_1},\dots,t^K_1,\dots,t^K_{N_K},a_0,\dots,a_K}$ as the probability measure having density with respect to the Lebesgue measure on $\mathcal{B}(\underbrace{\mathbb{R}^2 \times \cdots \times \mathbb{R}^2}_{1+n+K \text{ times}})$ that reads:

$$p_{SD}\left(x_{t^1_1},\dots,x_{t^1_{N_1}},\dots,x_{t^K_1},\dots,x_{t^K_{N_K}},x_{a_0},\dots,x_{a_K}\right) = p_b(x_{a_0},\dots,x_{a_K}) \times \prod_{k=1}^{K} \mathcal{N}\left(x_{t^k_1},\dots,x_{t^k_{N_k}} | x_{a_{k-1}}, x_{a_k}\right) \tag{C.12}$$

where p_b is as per Theorem 4.2 (A-1) and $\mathcal{N}\left(x_{t^k_1},\dots,x_{t^k_{N_k}} | x_{a_{k-1}}, x_{a_k}\right)$ is the (Gaussian) pdf of the joint distribution of the values at times $\{t^k_i \in]a_{k-1}, a_k[\}$ of the *conditional derivative Gaussian process* with unconditional mean functions m_k and unconditional covariance functions k_k that is conditioned to take values $x_{a_{k-1}} = (z_{a_{k-1}}, z'_{a_{k-1}})$ and $x_{a_k} = (z_{a_k}, z'_{a_k})$ at times a_{k-1} and a_k respectively (the corresponding—conditional—mean and covariance functions are derived from Equations (4.3) and (4.4)). Let us extend the family of measures ν^{SD} to cases where some or all boundary times a_k are missing, by integrating out the corresponding variables in Equation (C.12). For instance when a_0 and a_1 are missing,

$$\begin{aligned}&\nu^{SD}_{t^1_1,\dots,t^1_{N_1},\dots,t^K_1,\dots,t^K_{N_K},a_2,\dots,a_K}(T^1_1,\dots,T^1_{N_1},\dots,T^K_1,\dots,T^K_{N_K},A_2,\dots,A_K)\\ &:= \nu^{SD}_{t^1_1,\dots,t^1_{N_1},\dots,t^K_1,\dots,t^K_{N_K},a_0,\dots,a_K}(T^1_1,\dots,T^1_{N_1},\dots,T^K_1,\dots,T^K_{N_K},\mathbb{R}^2,\mathbb{R}^2,A_2,\dots,A_K)\end{aligned} \tag{C.13}$$

where A_i and T^i_j are rectangle in $\mathbb{R}^2$. Finally, we extend the family of measures ν^{SD} to any arbitrary set of indices $\{t_1,\dots,t_n\}$ as follows:

$$\nu^{SD}_{t_1,\dots,t_n}(T_1,\dots,T_n) := \nu^{SD}_{t_{\pi^*(1)},\dots,t_{\pi^*(n)}}(T_{\pi^*(1)},\dots,T_{\pi^*(n)}), \tag{C.14}$$

where π^* is a permutation of $\{1,\dots,n\}$ such that $\{t_{\pi^*(1)},\dots,t_{\pi^*(n)}\}$ verify the following conditions:

1. $\forall\ i,\ j,$ if $t_i \in]a_{k_1-1}, a_{k_1}[,\ t_j \in]a_{k_2-1}, a_{k_2}[,$ and $k_1 < k_2$, then $\mathrm{Idx}(t_i) < \mathrm{Idx}(t_j)$. Where $\mathrm{Idx}(t_i)$ stands for the index of t_i in $\{t_{\pi^*(1)},\dots,t_{\pi^*(n)}\}$;
2. if $t_i \notin \{a_0,\dots,a_K\}$ and $t_j \in \{a_0,\dots,a_K\}$ then $\mathrm{Idx}(t_i) < \mathrm{Idx}(t_j)$;

3. if $t_i \in \{a_0, \ldots, a_K\}$ and $t_j \in \{a_0, \ldots, a_K\}$ then $\text{Idx}(t_i) < \text{Idx}(t_j)$ if and only if $t_i < t_j$.

Any such measure $\nu^{SD}_{t_{\pi^*(1)},\ldots,t_{\pi^*(n)}}$ will fall in the category of either Equation (C.12) or Equation (C.13). Although π^* is not unique, any two permutations satisfying the above conditions will only differ by a permutation of times belonging to the same string interval $]a_{k-1}, a_k[$. Moreover, it follows from Equations (C.12) and (C.13) that the measures $\nu^{SD}_{t_{\pi^*(1)},\ldots,t_{\pi^*(n)}}$ are invariant by permutation of times belonging to the same string interval $]a_{k-1}, a_k[$, and as a result any two π^* satisfying the above conditions will yield the same probability measure.

The finite dimensional probability measures $\nu^{SD}_{t_1,\ldots,t_n}$ are the measures implied by Theorem 4.2. The permutation condition (C.1) of Kolmogorov's extension theorem is met by virtue of Equation (C.14). In effect for every permutation π of $\{1, \ldots, n\}$, if we let $\pi' : \{\pi(1), \ldots, \pi(n)\} \to \{\pi^*(1), \ldots, \pi^*(n)\}$, then

$$\begin{aligned}
\nu^{SD}_{t_{\pi(1)},\ldots,t_{\pi(n)}}(T_{\pi(1)}, \ldots, T_{\pi(n)}) &:= \nu^{SD}_{t_{\pi'\circ\pi(1)},\ldots,t_{\pi'\circ\pi(n)}}(T_{\pi'\circ\pi(1)}, \ldots, T_{\pi'\circ\pi(n)}) \\
&= \nu^{SD}_{t_{\pi^*(1)},\ldots,t_{\pi^*(n)}}(T_{\pi^*(1)}, \ldots, T_{\pi^*(n)}) \\
&= \nu^{SD}_{t_1,\ldots,t_n}(T_1, \ldots, T_n).
\end{aligned}$$

As for the marginalisation condition (C.2), it is met for every boundary time by virtue of how we extended ν^{SD} to missing boundary times. All we need to prove now is that the marginalisation condition is also met at any non-boundary time. To do so, it is sufficient to prove that the marginalisation condition holds for t^1_1, that is:

$$\begin{aligned}
&\nu^{SD}_{t^1_1,\ldots,t^1_{N_1},\ldots,t^K_1,\ldots,t^K_{N_K},a_0,\ldots,a_K}(\mathbb{R}^2, T^1_2, \ldots, T^1_{N_1}, \ldots, T^K_1, \ldots, T^K_{N_K}, A_0, \ldots, A_K) \\
&= \nu^{SD}_{t^1_2,\ldots,t^1_{N_1},\ldots,t^K_1,\ldots,t^K_{N_K},a_0,\ldots,a_K}(T^1_2, \ldots, T^1_{N_1}, \ldots, T^K_1, \ldots, T^K_{N_K}, A_0, \ldots, A_K)
\end{aligned} \tag{C.15}$$

for every rectangles A_i and T^i_j in $\mathbb{R}^2$. In effect, cases where some boundary times are missing are special cases with the corresponding rectangles A_j set to $\mathbb{R}^2$. Moreover, if we prove Equation (C.15), the permutation property (C.1) will allow us to conclude that the marginalisation also holds true for any other (single) non-boundary time. Furthermore, if Equation (C.15) holds true, it can be shown that the marginalisation condition will also hold over multiple non-boundary times by using the permutation property (C.1) and marginalising one non-boundary time after another.

By Fubini's theorem, and considering Equation (C.12), showing that Equation (C.15) holds true is equivalent to showing that:

$$\int_{\mathbb{R}^2} \mathcal{N}\left(x_{t_1^1}, \ldots, x_{t_{N_1}^1} | x_{a_0}, x_{a_1}\right) dx_{t_1^1} = \mathcal{N}\left(x_{t_2^1}, \ldots, x_{t_{N_1}^1} | x_{a_0}, x_{a_1}\right) \tag{C.16}$$

which holds true as $\mathcal{N}\left(x_{t_1^1}, \ldots, x_{t_{N_1}^1} | x_{a_0}, x_{a_1}\right)$ is a multivariate Gaussian density, and the corresponding marginal is indeed the density of the same *conditional derivative Gaussian process* at times $t_2^1, \ldots, t_{N_1}^1$.

This concludes the proof of the existence of the stochastic process $(SD_t)_{t\in I}$.

C.2.2 Proof of Theorem 4.2 (B)

As conditional on boundary conditions the restriction of a *string derivative Gaussian process* on a string interval $[a_{k-1}, a_k]$ is a *derivative Gaussian process*, it follows from Proposition 4.1 (C) that

$$\begin{aligned} &\forall\ \tilde{x}_{a_0}, \ldots, \tilde{x}_{a_K},\ \forall\ t, t+h \in [a_{k-1}, a_k], \\ &\lim_{h\to 0} \mathrm{E}\left(\left[\frac{z_{t+h} - z_t}{h} - z_t'\right]^2 \middle| x_{a_0} = \tilde{x}_{a_0}, \ldots, x_{a_K} = \tilde{x}_{a_K}\right) = 0, \end{aligned} \tag{C.17}$$

or equivalently that:

$$\Delta z_h := \mathrm{E}\left(\left[\frac{z_{t+h} - z_t}{h} - z_t'\right]^2 \middle| x_{a_0}, \ldots, x_{a_K}\right) \xrightarrow[h\to 0]{a.s.} 0. \tag{C.18}$$

Moreover,

$$\Delta z_h = \mathrm{Var}\left(\frac{z_{t+h} - z_t}{h} - z_t' \middle| x_{a_0}, \ldots, x_{a_K}\right) + \mathrm{E}\left(\frac{z_{t+h} - z_t}{h} - z_t' \middle| x_{a_0}, \ldots, x_{a_K}\right)^2. \tag{C.19}$$

As both terms in the sum of the above equation are non-negative, it follows that

$$\mathrm{Var}\left(\frac{z_{t+h} - z_t}{h} - z_t' \middle| x_{a_0}, \ldots, x_{a_K}\right) \xrightarrow[h\to 0]{a.s.} 0 \quad \text{and} \quad \mathrm{E}\left(\frac{z_{t+h} - z_t}{h} - z_t' \middle| x_{a_0}, \ldots, x_{a_K}\right)^2 \xrightarrow[h\to 0]{a.s.} 0.$$

From which we deduce

$$\mathrm{E}\left(\frac{z_{t+h} - z_t}{h} - z_t' \middle| x_{a_0}, \ldots, x_{a_K}\right) \xrightarrow[h\to 0]{a.s.} 0.$$

As $\mathrm{E}\left(\frac{z_{t+h}-z_t}{h} - z'_t \middle| x_{a_0}, \ldots, x_{a_K}\right)$ depends linearly on the boundary conditions, and as the boundary conditions are jointly-Gaussian (see C.7 step 1), it follows that $\mathrm{E}\left(\frac{z_{t+h}-z_t}{h} - z'_t \middle| x_{a_0}, \ldots, x_{a_K}\right)$ is Gaussian. Finally we note that

$$\mathrm{Var}\left(\frac{z_{t+h}-z_t}{h} - z'_t \middle| x_{a_0}, \ldots, x_{a_K}\right)$$

does not depend on the values of the boundary conditions x_{a_k} (but rather on the boundary times), and we recall that convergence almost sure of Gaussian random variables implies convergence in L^2. Hence, taking the expectation on both side of Equation (C.19) and then the limit as h goes to 0 we get

$$\mathrm{E}\left(\left[\frac{z_{t+h}-z_t}{h} - z'_t\right]^2\right) = \mathrm{E}(\Delta z_h) \underset{h\to 0}{\longrightarrow} 0,$$

which proves that the *string GP* $(z_t)_{t\in I}$ is differentiable in the L^2 sense on I and has derivative $(z'_t)_{t\in I}$.

We prove the continuity in the L^2 sense of $(z'_t)_{t\in I}$ in a similar fashion, noting that conditional on the boundary conditions, $(z'_t)_{t\in I}$ is a Gaussian process whose mean function $\frac{dm_{ck}^{a_{k-1},a_k}}{dt}$ and covariance function $\frac{\partial^2 k_{ck}^{a_{k-1},a_k}}{\partial x \partial y}$ are continuous, thus is continuous in the L^2 sense on $[a_{k-1}, a_k]$ (conditional on the boundary conditions). We therefore have that:

$$\forall\ \tilde{x}_{a_0}, \ldots, \tilde{x}_{a_K},\ \forall\ t, t+h \in [a_{k-1}, a_k],\ \lim_{h\to 0} \mathrm{E}\left((z'_{t+h} - z'_t)^2 \middle| x_{a_0} = \tilde{x}_{a_0}, \ldots, x_{a_K} = \tilde{x}_{a_K}\right) = 0, \tag{C.20}$$

from which we get that:

$$\Delta z'_h := \mathrm{E}\left(\left[z'_{t+h} - z'_t\right]^2 \middle| x_{a_0}, \ldots, x_{a_K}\right) \underset{h\to 0}{\overset{a.s.}{\longrightarrow}} 0. \tag{C.21}$$

Moreover,

$$\Delta z'_h = \mathrm{Var}\left(z'_{t+h} - z'_t \middle| x_{a_0}, \ldots, x_{a_K}\right) + \mathrm{E}\left(z'_{t+h} - z'_t \middle| x_{a_0}, \ldots, x_{a_K}\right)^2, \tag{C.22}$$

which implies that

$$\mathrm{Var}\left(z'_{t+h} - z'_t \middle| x_{a_0}, \ldots, x_{a_K}\right) \underset{h\to 0}{\overset{a.s.}{\longrightarrow}} 0$$

and

$$\mathrm{E}\left(z'_{t+h} - z'_t \middle| x_{a_0}, \ldots, x_{a_K}\right)^2 \underset{h\to 0}{\overset{a.s.}{\longrightarrow}} 0,$$

as both terms in the sum in Equation (C.22) are non-negative. Finally,

$$\mathrm{Var}\left(z'_{t+h} - z'_t \middle| x_{a_0}, \ldots, x_{a_K}\right)$$

does not depend on the values of the boundary conditions, and

$$\mathrm{E}\left(z'_{t+h} - z'_t \middle| x_{a_0}, \ldots, x_{a_K}\right)$$

is Gaussian for the same reason as before. Hence, taking the expectation on both sides of Equation (C.22) , we get that

$$\mathrm{E}\left(\left[z'_{t+h} - z'_t\right]^2\right) = \mathrm{E}(\Delta z'_h) \underset{h\to 0}{\longrightarrow} 0,$$

which proves that (z'_t) is continuous in the L^2 sense. ■

C.3 Proof of the Condition for Pathwise Regularity Upgrade of *String GPs* from L^2

In this section we prove that a sufficient condition for the process $(z'_t)_{t\in I}$ in Theorem 4.2 to be almost surely continuous and to be the almost sure derivative of the string Gaussian process $(z_t)_{t\in I}$, is that the Gaussian processes on $I_k = [a_{k-1}, a_k]$ with mean and covariance functions $m_{ck}^{a_{k-1},a_k}$ and $k_{ck}^{a_{k-1},a_k}$ (as per Equations (4.3) and (4.4) with $m := m_k$ and $k := k_k$) are themselves almost surely $\mathcal{C}^1$ for every boundary condition.

Firstly we note that the above condition guarantees that the result holds at non-boundary times. As for boundary times, the condition implies that the *string GP* is almost surely right differentiable (resp. left differentiable) at every left (resp. right) boundary time, including a_0 and a_K. Moreover, the *string GP* being differentiable in L^2, the right hand side and left hand side almost sure derivatives are the same, and are equal to the L^2 derivative, which proves that the L^2 derivatives at inner boundary times are also in the almost sure sense. A similar argument holds to conclude that the right (resp. left) hand side derivative at a_0 (resp. a_K) is also in the almost sure sense. Moreover, the derivative process $(z'_t)_{t\in I}$ admits an almost sure right hand side limit and an almost sure left hand side limit at every inner boundary time and both are equal as the derivative is continuous in L^2, which proves its almost sure continuity at

inner boundary times. Almost sure continuity of $(z'_t)_{t\in I}$ on the right (resp. left) of a_0 (resp. a_K) is a direct consequence of the above condition.

C.4 Proof of Proposition C.4

In this section, we prove Proposition C.4, which we recall below.

Proposition C.4 (Additively separable *string GPs* are flexible)
Let $k(x,y) := \rho\left(||x-y||^2_{L^2}\right)$ be a stationary covariance function generating a.s. $\mathcal{C}^1$ GP paths indexed on $\mathbb{R}^d$, $d>0$, and ρ a function that is $\mathcal{C}^2$ on $]0,+\infty[$ and continuous at 0. Let $\phi_s(x_1,\dots,x_d)=\sum_{j=1}^d x_j$, let $(z^j_t)_{t\in I^j,\ j\in[1..d]}$ be independent stationary Gaussian processes with mean 0 and covariance function k (where the L^2 norm is on $\mathbb{R}$), and let $f(t_1,\dots,t_d)=\phi_s(z^1_{t_1},\dots,z^d_{t_d})$ be the corresponding stationary string GP. Finally, let g be an isotropic Gaussian process indexed on $I^1\times\cdots\times I^d$ with mean 0 and covariance function k (where the L^2 norm is on $\mathbb{R}^d$). Then:
1) $\forall\ x\in I^1\times\cdots\times I^d,\ H(\nabla f(x)) = H(\nabla g(x))$,
2) $\forall\ x\neq y\in I^1\times\cdots\times I^d,\ I(\nabla f(x);\nabla f(y)) \leq I(\nabla g(x);\nabla g(y))$.

To prove Proposition C.4 we need a lemma which we state and prove below.

Lemma C.2 *Let X_n be a sequence of Gaussian random vectors with auto-covariance matrix Σ_n and mean μ_n, converging almost surely to X_∞. If $\Sigma_n\to\Sigma_\infty$ and $\mu_n\to\mu_\infty$ then X_∞ is Gaussian with mean μ_∞ and auto-covariance matrix Σ_∞.*

Proof We need to show that the characteristic function of X_∞ is

$$\phi_{X_\infty}(t) := \mathrm{E}(e^{itX_\infty}) = e^{it^T\mu_\infty-\frac{1}{2}t^T\Sigma_\infty t}.$$

As Σ_n is positive semi-definite, $\forall n,\ |e^{it^T\mu_n-\frac{1}{2}t^T\Sigma_n t}| = e^{-\frac{1}{2}t^T\Sigma_n t}\leq 1$. Hence, by Lebesgue's Dominated Convergence theorem,

$$\phi_{X_\infty}(t) = \mathrm{E}(\lim_{n\to+\infty} e^{it^TX_n}) = \lim_{n\to+\infty}\mathrm{E}(e^{it^TX_n}) = \lim_{n\to+\infty} e^{it^T\mu_n-\frac{1}{2}t^T\Sigma_n t} = e^{it^T\mu_\infty-\frac{1}{2}t^T\Sigma_\infty t}.$$

■

C.4.1 Proof of Proposition C.4 1)

Let $x = (t_1^x, \dots, t_d^x) \in I^1 \times \cdots \times I^d$. We want to show that $H(\nabla f(x)) = H(\nabla g(x))$ where f and g are as per Proposition C.4, and H is the entropy operator. Firstly, we note from Equation (4.11) that

$$\nabla f(x) = \left(z^{1\prime}_{t_1^x}, \dots, z^{d\prime}_{t_d^x}\right), \tag{C.23}$$

where the joint law of the GP $(z_t^j)_{t\in I^j}$ and its derivative $(z_t^{j\prime})_{t\in I^j}$ is provided in Proposition 4.1. As the processes $\left(z_t^j, z_t^{j\prime}\right)_{t\in I^j}$, $j \in [1..d]$ are assumed to be independent of each other, $\nabla f(x)$ is a Gaussian vector and its covariance matrix reads:

$$\Sigma_{\nabla f(x)} = -2\frac{\mathrm{d}\rho}{\mathrm{d}x}(0)\mathrm{I}_d, \tag{C.24}$$

where I_d is the $d \times d$ identity matrix. Hence,

$$H(\nabla f(x)) = \frac{d}{2}\left(1 + \ln(2\pi)\right) + \frac{1}{2}\ln|\Sigma_{\nabla f(x)}|. \tag{C.25}$$

Secondly, let e_j denote the d-dimensional vector whose j-th coordinate is 1 and every other coordinate is 0, and let $h \in \mathbb{R}$. As the proposition assumes the covariance function k generates almost surely $\mathcal{C}^1$ surfaces, the vectors $\left(\frac{g(x+he_1)-g(x)}{h}, \dots, \frac{g(x+he_d)-g(x)}{h}\right)$ are Gaussian vectors converging almost surely as $h \to 0$. Moreover, their mean is 0 and their covariance matrices have as element on the i-th row and j-th column ($i \neq j$):

$$\operatorname{cov}\left(\frac{g(x+he_i)-g(x)}{h}, \frac{g(x+he_j)-g(x)}{h}\right) = \frac{\rho(2h^2) - 2\rho(h^2) + \rho(0)}{h^2} \tag{C.26}$$

and as diagonal terms:

$$\operatorname{Var}\left(\frac{g(x+he_j)-g(x)}{h}\right) = 2\frac{\rho(0) - \rho(h^2)}{h^2}. \tag{C.27}$$

Taking the limit of Equations (C.26) and (C.27) using the first order Taylor expansion of ρ (which the Proposition assumes is $\mathcal{C}^2$), we get that:

$$\Sigma_{\nabla g(x)} = -2\frac{\mathrm{d}\rho}{\mathrm{d}x}(0)\mathrm{I}_d = \Sigma_{\nabla f(x)}, \tag{C.28}$$

It then follows from Lemma C.2 that the limit $\nabla g(x)$ of

$$\left(\frac{g(x+he_1)-g(x)}{h},\dots,\frac{g(x+he_d)-g(x)}{h}\right)$$

is also a Gaussian vector, which proves that $H(\nabla f(x)) = H(\nabla g(x))$.

C.4.2 Proof of Proposition C.4 2)

We start by stating and proving another lemma we will later use.

Lemma C.3 *Let A and B be two d-dimensional jointly Gaussian vectors with diagonal covariance matrices Σ_A and Σ_B respectively. Let $\Sigma_{A,B}$ be the cross-covariance matrix between A and B, and let $diag(\Sigma_{A,B})$ be the diagonal matrix whose diagonal is that of $\Sigma_{A,B}$. Then:*

$$det\left(\begin{bmatrix}\Sigma_A & diag(\Sigma_{A,B})\\ diag(\Sigma_{A,B}) & \Sigma_B\end{bmatrix}\right) \geq det\left(\begin{bmatrix}\Sigma_A & \Sigma_{A,B}\\ \Sigma_{A,B}^T & \Sigma_B\end{bmatrix}\right).$$

Proof Firstly we note that

$$\det\left(\begin{bmatrix}\Sigma_A & \mathrm{diag}(\Sigma_{A,B})\\ \mathrm{diag}(\Sigma_{A,B}) & \Sigma_B\end{bmatrix}\right) = \det(\Sigma_A)\det\left(\Sigma_B - \mathrm{diag}(\Sigma_{A,B})\Sigma_A^{-1}\mathrm{diag}(\Sigma_{A,B})\right)$$

and

$$\det\left(\begin{bmatrix}\Sigma_A & \Sigma_{A,B}\\ \Sigma_{A,B} & \Sigma_B\end{bmatrix}\right) = \det(\Sigma_A)\det\left(\Sigma_B - \Sigma_{A,B}^T\Sigma_A^{-1}\Sigma_{A,B}\right).$$

As the matrix Σ_A is positive semi-definite, $\det(\Sigma_A) \geq 0$. The case $\det(\Sigma_A) = 0$ is straight-forward. Thus we assume that $\det(\Sigma_A) > 0$, so that all we need to prove is that

$$\det\left(\Sigma_B - \mathrm{diag}(\Sigma_{A,B})\Sigma_A^{-1}\mathrm{diag}(\Sigma_{A,B})\right) \geq \det\left(\Sigma_B - \Sigma_{A,B}^T\Sigma_A^{-1}\Sigma_{A,B}\right).$$

Secondly, the matrix $\Sigma_{B|A}^{\mathrm{diag}} := \Sigma_B - \mathrm{diag}(\Sigma_{A,B})\Sigma_A^{-1}\mathrm{diag}(\Sigma_{A,B})$ being diagonal, its determinant is the product of its diagonal terms:

$$\det(\Sigma_{B|A}^{\mathrm{diag}}) = \prod_{i=1}^{d}\Sigma_{B|A}^{\mathrm{diag}}[i,i] = \prod_{i=1}^{d}\left(\Sigma_B[i,i] - \frac{\Sigma_{A,B}[i,i]^2}{\Sigma_A[i,i]}\right).$$

As for the matrix $\Sigma_{B|A} := \Sigma_B - \Sigma_{A,B}^T\Sigma_A^{-1}\Sigma_{A,B}$, we note that it happens to be the covariance matrix of the (Gaussian) distribution of B given A, and thus is positive

semi-definite and admits a Cholesky decomposition $\Sigma_{B|A} = LL^T$. It follows that

$$\det(\Sigma_{B|A}) = \prod_{i=1}^{d} L[i,i]^2 \leq \prod_{i=1}^{d} \Sigma_{B|A}[i,i] = \prod_{i=1}^{d} \left(\Sigma_B[i,i] - \sum_{j=1}^{d} \frac{\Sigma_{A,B}[j,i]^2}{\Sigma_A[j,j]} \right)$$
$$\leq \prod_{i=1}^{d} \left(\Sigma_B[i,i] - \frac{\Sigma_{A,B}[i,i]^2}{\Sigma_A[i,i]} \right) = \det(\Sigma^{\text{diag}}_{B|A}), \tag{C.29}$$

where the first inequality results from the fact that $\Sigma_{B|A}[i,i] = \sum_{j=1}^{i} L[j,i]^2$ by definition of the Cholesky decomposition. This proves that

$$\det\left(\Sigma_B - \text{diag}(\Sigma_{A,B})\Sigma_A^{-1}\text{diag}(\Sigma_{A,B})\right) \geq \det\left(\Sigma_B - \Sigma_{A,B}^T\Sigma_A^{-1}\Sigma_{A,B}\right),$$

which as previously discussed concludes the proof of the lemma. ■

Proof of Proposition C.4 2): Let $x = (t_1^x, \ldots, t_d^x),\ y = (t_1^y, \ldots, t_d^y) \in I^1 \times \cdots \times I^d$, $x \neq y$. We want to show that $I(\nabla f(x); \nabla f(y)) \leq I(\nabla g(x); \nabla g(y))$ where f and g are as per Proposition C.4, and

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

is the mutual information between X and Y. As we have proved that $\forall\, x,\ H(\nabla f(x)) = H(\nabla g(x))$, all we need to prove now is that

$$H(\nabla f(x), \nabla f(y)) \geq H(\nabla g(x), \nabla g(y)).$$

Firstly, it follows from Equation (C.23) and the fact that the *derivative Gaussian processes* $(z_t^j, z_t^{j'})_{t \in I^j}$ are independent that $(\nabla f(x), \nabla f(y))$ is a jointly Gaussian vector. Moreover, the cross-covariance matrix $\Sigma_{\nabla f(x), \nabla f(y)}$ is diagonal with diagonal terms:

$$\Sigma_{\nabla f(x), \nabla f(y)}[i,i] = -2\left[\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(||x-y||^2_{L^2}\right) + 2(t_i^x - t_i^y)^2 \frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(||x-y||^2_{L^2}\right)\right]. \tag{C.30}$$

Secondly, it follows from a similar argument to the previous proof that $(\nabla g(x), \nabla g(y))$ is also a jointly Gaussian vector, and the terms $\Sigma_{\nabla g(x), \nabla g(y)}[i,j]$ are evaluated as limit of the cross-covariance terms $\text{cov}\left(\frac{g(x+he_i)-g(x)}{h}, \frac{g(y+he_j)-g(y)}{h}\right)$ as $h \to 0$. For $i = j$,

$$\text{cov}\left(\frac{g(x+he_i)-g(x)}{h}, \frac{g(y+he_i)-g(y)}{h}\right) = \frac{1}{h^2}\left\{2\rho\left(\sum_k (t_k^x - t_k^y)^2\right)\right.$$

$$- \rho\left(\sum_{k\neq i}(t_k^x - t_k^y)^2 + (t_i^x + h - t_i^y)^2\right) - \rho\left(\sum_{k\neq i}(t_k^x - t_k^y)^2 + (t_i^x - h - t_i^y)^2\right)\Big\}, \quad \text{(C.31)}$$

As ρ is assumed to be $\mathcal{C}^2$, the below Taylor expansions around $h = 0$ hold true:

$$\rho\left(\sum_{k}(t_k^x - t_k^y)^2\right) - \rho\left(\sum_{k\neq i}(t_k^x - t_k^y)^2 + (t_i^x - h - t_i^y)^2\right) = 2(t_i^x - t_i^y)h\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_{k}(t_k^x - t_k^y)^2\right) \quad \text{(C.32)}$$

$$- \left[\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_{k}(t_k^x - t_k^y)^2\right) + 2(t_i^x - t_i^y)^2\frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(\sum_{k}(t_k^x - t_k^y)^2\right)\right]h^2 + o(h^2)$$

$$\rho\left(\sum_{k}(t_k^x - t_k^y)^2\right) - \rho\left(\sum_{k\neq i}(t_k^x - t_k^y)^2 + (t_i^x + h - t_i^y)^2\right) = -2(t_i^x - t_i^y)h\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_{k}(t_k^x - t_k^y)^2\right) \quad \text{(C.33)}$$

$$- \left[\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_{k}(t_k^x - y_k)^2\right) + 2(t_i^x - t_i^y)^2\frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(\sum_{k}(t_k^x - t_k^y)^2\right)\right]h^2 + o(h^2)$$

Plugging Equations (C.32) and (C.33) into Equation (C.31) and taking the limit we obtain:

$$\begin{aligned}\Sigma_{\nabla g(x),\nabla g(y)}[i,i] &= -2\left[\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(||x-y||_{L^2}^2\right) + 2(t_i^x - t_i^y)^2\frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(||x-y||_{L^2}^2\right)\right] \\ &= \Sigma_{\nabla f(x),\nabla f(y)}[i,i]. \end{aligned} \quad \text{(C.34)}$$

Similarly for $i \neq j$,

$$\begin{aligned}\mathrm{cov}\left(\frac{g(x+he_i)-g(x)}{h}, \frac{g(y+he_j)-g(y)}{h}\right) &= \frac{1}{h^2}\Big\{\rho\Big(\sum_{k\neq i,j}(t_k^x - t_k^y)^2 + (t_i^x + h - t_i^y)^2 \\ &+ (t_j^x - h - t_j^y)^2\Big) - \rho\left(\sum_{k\neq i}(t_k^x - t_k^y)^2 + (t_i^x + h - t_i^y)^2\right) - \rho\left(\sum_{k\neq j}(t_k^x - t_k^y)^2 + (t_i^x - h - t_j^y)^2\right) \\ &+ \rho\left(\sum_{k}(t_k^x - t_k^y)^2\right)\Big\}, \end{aligned} \quad \text{(C.35)}$$

and

$$\rho\left(\sum_{k\neq i,j}(t_k^x - t_k^y)^2 + (t_i^x + h - t_i^y)^2 + (t_j^x - h - t_j^y)^2\right) - \rho\left(\sum_{k}(t_k^x - t_k^y)^2\right)$$

$$= \left[2\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_k (t_k^x - t_k^y)^2\right) + 2\left((t_i^x - t_i^y) - (t_j^x - t_j^y)\right)^2 \times \frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(\sum_k (t_k^x - t_k^y)^2\right)\right] h^2$$
$$+ 2\left(t_i^x - t_i^y - t_j^x + t_j^y\right)\frac{\mathrm{d}\rho}{\mathrm{d}x}\left(\sum_k (t_k^x - t_k^y)^2\right) h + o(h^2). \tag{C.36}$$

Plugging Equations (C.32), (C.33) and (C.36) in Equation (C.35) and taking the limit we obtain:

$$\Sigma_{\nabla g(x),\nabla g(y)}[i,j] = -4(t_i^x - t_i^y)(t_j^x - t_j^y)\frac{\mathrm{d}^2\rho}{\mathrm{d}x^2}\left(||x-y||_{L^2}^2\right). \tag{C.37}$$

To summarize, $(\nabla f(x), \nabla f(y))$ and $(\nabla g(x), \nabla g(y))$ are both jointly Gaussian vectors; $\nabla f(x)$, $\nabla g(x)$, $\nabla f(y)$, and $\nabla g(y)$ are (Gaussian) identically distributed with a diagonal covariance matrix; $\Sigma_{\nabla f(x),\nabla f(y)}$ is diagonal; $\Sigma_{\nabla g(x),\nabla g(y)}$ has the same diagonal as $\Sigma_{\nabla f(x),\nabla f(y)}$ but has possibly non-zero off-diagonal terms. Hence, it follows from Lemma C.3 that the determinant of the auto-covariance matrix of $(\nabla f(x), \nabla f(y))$ is higher than that of the auto-covariance matrix of $(\nabla g(x), \nabla g(y))$; or equivalently the entropy of $(\nabla f(x), \nabla f(y))$ is higher than that of $(\nabla g(x), \nabla g(y))$ (as both are Gaussian vectors), which as previously discussed is sufficient to conclude that the mutual information between $\nabla f(x)$ and $\nabla f(y)$ is smaller than that between $\nabla g(x)$ and $\nabla g(y)$.

C.5 Proof of Proposition 4.6

In this section, we prove Proposition 4.6, which we recall below.

Proposition 4.6 (Extension of the *standard GP paradigm*)
Let $K \in \mathbb{N}^*$, let $I = [a_0, a_K]$ and $I_k = [a_{k-1}, a_k]$ be intervals with $a_0 < \cdots < a_K$. Furthermore, let $m : I \to \mathbb{R}$ be a $\mathcal{C}^1$ function, m_k the restriction of m to I_k, $h : I \times I \to \mathbb{R}$ a $\mathcal{C}^3$ symmetric positive semi-definite function, and h_k the restriction of h to $I_k \times I_k$. If

$$(z_t)_{t\in I} \sim \mathcal{SGP}(\{a_k\}, \{m_k\}, \{h_k\}),$$

then

$$\forall\ k \in [1..K],\ \ (z_t)_{t\in I_k} \sim \mathcal{GP}(m, h).$$

Proof
To prove Proposition 4.6, we consider the *string derivative Gaussian process* (Theorem 4.2) $(SD_t)_{t\in I}$, $SD_t = (z_t, z_t')$ with unconditional string mean and covariance functions

as per Proposition 4.6 and prove that its restrictions on the intervals $I_k = [a_{k-1}, a_k]$ are *derivative Gaussian processes* with the same mean function m and covariance function h. Proposition 4.1(B) will then allow us to conclude that $(z_t)_{t\in I_k}$ are GPs with mean m and covariance function h.

Let $t_1, \ldots, t_n \in]a_{k-1}, a_k[$ and let $p_D(x_{a_{k-1}})$ (respectively $p_D(x_{a_k}|x_{a_{k-1}})$ and $p_D(x_{t_1}, \ldots, x_{t_n}|x_{a_{k-1}}, x_{a_k})$) denote the pdf of the value of the *derivative Gaussian process* with mean function m and covariance function h at a_{k-1} (respectively its value at a_k conditional on its value at a_{k-1}, and its values at $t_1, \ldots, t_n$ conditional on its values at a_{k-1} and a_k). Saying that the restriction of the *string derivative Gaussian process* (SD_t) on $[a_{k-1}, a_k]$ is the *derivative Gaussian process* with mean m and covariance h is equivalent to saying that all finite dimensional marginals of the *string derivative Gaussian process* $p_{SD}(x_{a_{k-1}}, x_{t_1}, \ldots, x_{t_n}, x_{a_k})$, $t_i \in [a_{k-1}, a_k]$, factorise as[2]:

$$p_{SD}(x_{a_{k-1}}, x_{t_1}, \ldots, x_{t_n}, x_{a_k}) = p_D(x_{a_{k-1}})p_D(x_{a_k}|x_{a_{k-1}})p_D(x_{t_1}, \ldots, x_{t_n}|x_{a_{k-1}}, x_{a_k}).$$

Moreover, we know from Theorem 4.2 that by design, $p_{SD}(x_{a_{k-1}}, x_{t_1}, \ldots, x_{t_n}, x_{a_k})$ factorises as

$$p_{SD}(x_{a_{k-1}}, x_{t_1}, \ldots, x_{t_n}, x_{a_k}) = p_{SD}(x_{a_{k-1}})p_D(x_{a_k}|x_{a_{k-1}})p_D(x_{t_1}, \ldots, x_{t_n}|x_{a_{k-1}}, x_{a_k}).$$

In other words, all we need to prove is that

$$p_{SD}(x_{a_k}) = p_D(x_{a_k})$$

for every boundary time, which we will do by induction. We note by integrating out every boundary condition but the first in p_b (as per Theorem 4.2 (a-1)) that

$$p_{SD}(x_{a_0}) = p_D(x_{a_0}).$$

If we assume that $p_{SD}(x_{a_{k-1}}) = p_D(x_{a_{k-1}})$ for some $k > 0$, then as previously discussed the restriction of the *string derivative Gaussian process* on $[a_{k-1}, a_k]$ will be the *derivative Gaussian process* with the same mean and covariance functions, which will imply that $p_{SD}(x_{a_k}) = p_D(x_{a_k})$. This concludes the proof. ■

[2] We emphasize that the terms on the right hand-side of this equation involve p_D not p_{SD}.

C.6 Proof of Lemma C.4

In this section, we will prove Lemma C.4 that we recall below.

Lemma C.4 Let X be a multivariate Gaussian with mean μ_X and covariance matrix Σ_X. If conditional on X, Y is a multivariate Gaussian with mean $MX + A$ and covariance matrix Σ_Y^c where M, A and Σ_Y^c do not depend on X, then (X, Y) is a jointly Gaussian vector with mean

$$\mu_{X;Y} = \begin{bmatrix} \mu_X \\ M\mu_X + A \end{bmatrix},$$

and covariance matrix

$$\Sigma_{X;Y} = \begin{bmatrix} \Sigma_X & \Sigma_X M^T \\ M\Sigma_X & \Sigma_Y^c + M\Sigma_X M^T \end{bmatrix}.$$

Proof To prove this lemma we introduce two vectors $\tilde{X}$ and $\tilde{Y}$ whose lengths are the same as those of X and Y respectively, and such that $(\tilde{X}, \tilde{Y})$ is jointly Gaussian with mean $\mu_{X;Y}$ and covariance matrix $\Sigma_{X;Y}$. We then prove that the (marginal) distribution of $\tilde{X}$ is the same as the distribution of X and that the distribution of $\tilde{Y}|\tilde{X} = x$ is the same as $Y|X = x$ for any x, which is sufficient to conclude that (X, Y) and $(\tilde{X}, \tilde{Y})$ have the same distribution.

It is obvious from the joint $(\tilde{X}, \tilde{Y})$ that $\tilde{X}$ is Gaussian distribution with mean μ_X and covariance matrix Σ_X. As for the distribution of $\tilde{Y}$ conditional on $\tilde{X} = x$, it follows from the usual Gaussian identities that it is Gaussian with mean

$$M\mu_X + c + M\Sigma_X\Sigma_X^{-1}(x - \mu_X) = Mx + c,$$

and covariance matrix

$$\Sigma_Y^c + M\Sigma_X M^T - M\Sigma_X\Sigma_X^{-1}\Sigma_X^T M^T = \Sigma_Y^c,$$

which is the same distribution as that of $Y|X = x$ since the covariance matrix Σ_X is symmetric. This concludes our proof. ■

C.7 Proof of Proposition 4.5

In this section we will prove that *string GPs* with link function ϕ_s are GPs, or in other words that if f is a *string GP* indexed on $\mathbb{R}^d$, $d > 0$ with link function $\phi_s(x_1, \ldots, x_d) = \sum_{j=1}^d x_j$, then $(f(x_1), \ldots, f(x_n))$ has a multivariate Gaussian distribution for every set of distinct points $x_1, \ldots, x_n \in \mathbb{R}^d$.

Proof As the sum of independent Gaussian processes is a Gaussian process, a sufficient condition for additively separable *string GPs* to be GPs in dimensions $d > 1$ is that *string GPs* be GPs in dimension 1. Hence, all we need to do is to prove that *string GPs* are GPs in dimension 1.

Let $(z_t^j, z_t^{j\prime})_{t \in I^j}$ be a string derivative GP in dimension 1, with boundary times $a_0^j, \ldots, a_{K^j}^j$, and unconditional string mean and covariance functions m_k^j and k_k^j respectively. We want to prove that $(z_{t_1}^j, \ldots, z_{t_n}^j)$ is jointly Gaussian for any $t_1, \ldots, t_n \in I^j$.

Step 1 $(z_{a_0}^j, z_{a_0}^{j\prime}, \ldots, z_{a_{K^j}}^j, z_{a_{K^j}}^{j\prime})$ **is jointly Gaussian**

We first prove recursively that the vector $(z_{a_0}^j, z_{a_0}^{j\prime}, \ldots, z_{a_{K^j}}^j, z_{a_{K^j}}^{j\prime})$ is jointly Gaussian. We note from Theorem 4.2 that $(z_t^j, z_t^{j\prime})_{t \in [a_0, a_1]}$ is the *derivative Gaussian process* with mean m_1^j and covariance function k_1^j. Hence, $(z_{a_0}^j, z_{a_0}^{j\prime}, z_{a_1}^j, z_{a_1}^{j\prime})$ is jointly Gaussian. Moreover, let us assume that $\mathcal{B}_{k-1} := (z_{a_0}^j, z_{a_0}^{j\prime}, \ldots, z_{a_{k-1}}^j, z_{a_{k-1}}^{j\prime})$ is jointly Gaussian for some $k > 1$. Conditional on $\mathcal{B}_{k-1}$, $(z_{a_k}^j, z_{a_k}^{j\prime})$ is Gaussian with covariance matrix independent of $\mathcal{B}_{k-1}$, and with mean

$$\begin{bmatrix} m_k^j(a_k^j) \\ \frac{dm_k^j}{dt}(a_k^j) \end{bmatrix} + {}_k^j\mathbf{K}_{a_k^j; a_{k-1}^j} \, {}_k^j\mathbf{K}^{-1}_{a_{k-1}; a_{k-1}^j} \begin{bmatrix} z^j_{a_{k-1}^j} - m_k^j(a_{k-1}^j) \\ z^{j\prime}_{a_{k-1}^j} - \frac{dm_k^j}{dt}(a_{k-1}^j) \end{bmatrix},$$

which depends linearly on $(z_{a_0}^j, z_{a_0}^{j\prime}, \ldots, z_{a_{k-1}}^j, z_{a_{k-1}}^{j\prime})$. Hence by Lemma C.4,

$$(z_{a_0}^j, z_{a_0}^{j\prime}, \ldots, z_{a_k}^j, z_{a_k}^{j\prime})$$

is jointly Gaussian.

Step 2 $(z_{a_0}^j, z_{a_0}^{j\prime}, \ldots, z_{a_{K^j}}^j, z_{a_{K^j}}^{j\prime}, \ldots, z_{t_i^k}^j, z_{t_i^k}^{j\prime}, \ldots)$ **is jointly Gaussian**

Let $t_1^k, \ldots, t_{n^k}^k \in]a_{k-1}^j, a_k^j[$, $k \leq K^j$ be distinct string times. We want to prove that the vector $(z_{a_0}^j, z_{a_0}^{j\prime}, \ldots, z_{a_{K^j}}^j, z_{a_{K^j}}^{j\prime}, \ldots, z_{t_i^k}^j, z_{t_i^k}^{j\prime}, \ldots)$ where all boundary times are repre-

sented, and for any finite number of string times is jointly Gaussian. Firstly, we have already proved that $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{Kj}}, z^{j\prime}_{a_{Kj}})$ is jointly Gaussian. Secondly, we note from Theorem 4.2 that conditional on $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{Kj}}, z^{j\prime}_{a_{Kj}})$, $(\ldots, z^j_{t^k_i}, z^{j\prime}_{t^k_i}, \ldots)$ is a Gaussian vector whose covariance matrix does not depend on $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{Kj}}, z^{j\prime}_{a_{Kj}})$, and whose mean depends linearly on $\left(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{Kj}}, z^{j\prime}_{a_{Kj}}\right)$. Hence,

$$\left(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{Kj}}, z^{j\prime}_{a_{Kj}}, \ldots, z^j_{t^k_i}, z^{j\prime}_{t^k_i}, \ldots\right)$$

is jointly Gaussian (by Lemma C.4).

Step 3 $(z^j_{t_1}, \ldots, z^j_{t_n})$ is jointly Gaussian

$(z^j_{t_1}, z^{j\prime}_{t_1}, \ldots, z^j_{t_n}, z^{j\prime}_{t_n})$ is jointly Gaussian as it can be regarded as the marginal of some joint distribution of the form $(z^j_{a_0}, z^{j\prime}_{a_0}, \ldots, z^j_{a_{Kj}}, z^{j\prime}_{a_{Kj}}, \ldots, z^j_{t^k_i}, z^{j\prime}_{t^k_i}, \ldots)$. Hence, its marginal $(z^j_{t_1}, \ldots, z^j_{t_n})$ is also jointly Gaussian, which concludes our proof. ■

C.8 Derivation of Global String GP Mean and Covariance Functions

We begin with *derivative string GPs* indexed on $\mathbb{R}$. Extensions to *membrane GPs* are easily achieved for a broad range of link functions. In our exposition, we focus on the class of *elementary symmetric polynomials* (Macdonald (1995)). In addition to containing the link function ϕ_s previously introduced, this family of polynomials yields global covariance structures that have many similarities with existing kernel approaches, which we discuss in section 4.4.3.

For $n \leq d$, the n-th order elementary symmetric polynomial is given by

$$e_0(x_1, \ldots, x_d) := 1, \quad \forall 1 \leq n \leq d \;\; e_n(x_1, \ldots, x_d) = \sum_{1 \leq j_1 < j_2 < \cdots < j_n \leq d} x_{j_1} \ldots x_{j_n}. \quad \text{(C.38)}$$

As an illustration,

$$e_1(x_1, \ldots, x_d) = \sum_{j=1}^{d} x_j = \phi_s(x_1, \ldots, x_d),$$

$$e_2(x_1, \ldots, x_d) = x_1x_2 + x_1x_3 + \cdots + x_1x_d + \cdots + x_{d-1}x_d,$$

$$\ldots$$

$$e_d(x_1, \ldots, x_d) = \prod_{j=1}^{d} x_j = \phi_p(x_1, \ldots, x_d).$$

Let f denote a *membrane GP* indexed on $\mathbb{R}^d$ with link function e_n and by $(z_t^1), \ldots, (z_t^d)$ its independent building block *string GPs*. Furthermore, let m_k^j and k_k^j denote the unconditional mean and covariance functions corresponding to the k-th string of (z_t^j) defined on $[a_{k-1}^j, a_k^j]$. Finally, let us define

$$\boxed{\bar{m}^j(t) := \mathrm{E}(z_t^j), \quad \bar{m}^{j\prime}(t) := \mathrm{E}(z_t^{j\prime}),}$$

the global mean functions of the j-th building bloack *string GP* and of its derivative, where $\forall t \in I^j$. It follows from the independence of the building block *string GPs* (z_t^j) that:

$$\boxed{\bar{m}^f(t_1, \ldots, t_d) := \mathrm{E}(f(t_1, \ldots, t_d)) = e_n(\bar{m}^1(t_1), \ldots, \bar{m}^d(t_d)).}$$

Moreover, noting that

$$\frac{\partial e_n}{\partial x_j} = e_{n-1}(x_1, \ldots, x_{j-1}, x_{j+1}, \ldots, x_d),$$

it follows that:

$$\boxed{\bar{m}^{\nabla f}(t_1, \ldots, t_d) := \mathrm{E}\left(\nabla f(t_1, \ldots, t_d)\right) = \begin{bmatrix} \bar{m}^{1\prime}(t_1) e_{n-1}\left(\bar{m}^2(t_2), \ldots, \bar{m}^d(t_d)\right) \\ \ldots \\ \bar{m}^{d\prime}(t_d) e_{n-1}\left(\bar{m}^1(t_1), \ldots, \bar{m}^{d-1}(t_{d-1})\right) \end{bmatrix}.}$$

Furthermore, for any $u_j, v_j \in I^j$ we also have that

$$\boxed{\mathrm{cov}\left(f(u_1, \ldots, u_d), f(v_1, \ldots, v_d)\right) = e_n\left(\mathrm{cov}(z_{u_1}^1, z_{v_1}^1), \ldots, \mathrm{cov}(z_{u_d}^d, z_{v_d}^d)\right),}$$

$$\boxed{\mathrm{cov}\left(\frac{\partial f}{\partial x_i}(u_1, \ldots, u_d), f(v_1, \ldots, v_d)\right) = e_n\left(\mathrm{cov}(z_{u_1}^1, z_{v_1}^1), \ldots, \mathrm{cov}(z_{u_i}^{i\prime}, z_{v_i}^i), \ldots, \mathrm{cov}(z_{u_d}^d, z_{v_d}^d)\right),}$$

and for $i \leq j$

$$\operatorname{cov}\left(\frac{\partial f}{\partial x_i}(u_1,\ldots,u_d), \frac{\partial f}{\partial x_j}(v_1,\ldots,v_d)\right) = \begin{cases} e_n\left(\operatorname{cov}(z^1_{u_1}, z^1_{v_1}), \ldots, \operatorname{cov}(z^{i\prime}_{u_i}, z^i_{v_i}), \ldots, \operatorname{cov}(z^{j\prime}_{u_j}, z^j_{v_j}), \ldots, \operatorname{cov}(z^d_{u_d}, z^d_{v_d})\right), & \text{if } i < j \\ e_n\left(\operatorname{cov}(z^1_{u_1}, z^1_{v_1}), \ldots, \operatorname{cov}(z^{i\prime}_{u_i}, z^{i\prime}_{v_i}), \ldots, \operatorname{cov}(z^d_{u_d}, z^d_{v_d})\right), & \text{if } i = j \end{cases}.$$

Overall, for any elementary symmetric polynomial link function, multivariate mean and covariance functions are easily deduced from the previously boxed equations and the univariate quantities

$$\bar{m}^j(u), \ \bar{m}^{j\prime}(u), \ \text{and } {}^j\bar{\mathbf{K}}_{u;v} := \begin{bmatrix} \operatorname{cov}(z^j_u, z^j_v) & \operatorname{cov}(z^j_u, z^{j\prime}_v) \\ \operatorname{cov}(z^{j\prime}_u, z^j_v) & \operatorname{cov}(z^{j\prime}_u, z^{j\prime}_v) \end{bmatrix} = {}^j\bar{\mathbf{K}}^T_{v;u},$$

which we now derive. In this regards, we will need the following lemma.

Lemma C.4 *Let X be a multivariate Gaussian with mean μ_X and covariance matrix Σ_X. If conditional on X, Y is a multivariate Gaussian with mean $MX + A$ and covariance matrix Σ^c_Y where M, A and Σ^c_Y do not depend on X, then (X, Y) is a jointly Gaussian vector with mean*

$$\mu_{X;Y} = \begin{bmatrix} \mu_X \\ M\mu_X + A \end{bmatrix},$$

and covariance matrix

$$\Sigma_{X;Y} = \begin{bmatrix} \Sigma_X & \Sigma_X M^T \\ M\Sigma_X & \Sigma^c_Y + M\Sigma_X M^T \end{bmatrix}.$$

Proof See Appendix C.6. ■

Global String GP Mean Functions

We now turn to evaluating the univariate global mean functions $\bar{m}^j$ and $\bar{m}^{j\prime}$. We start with boundary times and then generalise to other times.

Boundary times: We note from Theorem 4.2 that the restriction $\left(z^j_t, z^{j\prime}_t\right)_{t\in[a^j_0, a^j_1]}$ is the *derivative Gaussian process* with mean and covariance functions m^j_1 and k^j_1.

Thus,

$$\boxed{\begin{bmatrix}\bar{m}^j(a_0^j)\\ \bar{m}^{j\prime}(a_0^j)\end{bmatrix} = \begin{bmatrix}m_1^j(a_0^j)\\ \frac{dm_1^j}{dt}(a_0^j)\end{bmatrix}, \quad \text{and} \quad \begin{bmatrix}\bar{m}^j(a_1^j)\\ \bar{m}^{j\prime}(a_1^j)\end{bmatrix} = \begin{bmatrix}m_1^j(a_1^j)\\ \frac{dm_1^j}{dt}(a_1^j)\end{bmatrix}.}$$

For $k > 1$, we recall that conditional on $\left(z^j_{a^j_{k-1}}, z^{j\prime}_{a^j_{k-1}}\right)$, $\left(z^j_{a^j_k}, z^{j\prime}_{a^j_k}\right)$ is Gaussian with mean

$$\begin{bmatrix}m_k^j(a_k^j)\\ \frac{dm_k^j}{dt}(a_k^j)\end{bmatrix} + {}^j_k\mathbf{K}_{a^j_k;a^j_{k-1}}\, {}^j_k\mathbf{K}^{-1}_{a^j_{k-1};a^j_{k-1}} \begin{bmatrix}z^j_{a^j_{k-1}} - m_k^j(a_{k-1}^j)\\ z^{j\prime}_{a^j_{k-1}} - \frac{dm_k^j}{dt}(a_{k-1}^j)\end{bmatrix},$$

$$\text{with } {}^j_k\mathbf{K}_{u;v} = \begin{bmatrix}k_k^j(u,v) & \frac{\partial k_k^j}{\partial y}(u,v)\\ \frac{\partial k_k^j}{\partial x}(u,v) & \frac{\partial^2 k_k^j}{\partial x \partial y}(u,v)\end{bmatrix}.$$

It then follows from the law of total expectations that for all $k > 1$

$$\boxed{\begin{bmatrix}\bar{m}^j(a_k^j)\\ \bar{m}^{j\prime}(a_k^j)\end{bmatrix} = \begin{bmatrix}m_k^j(a_k^j)\\ \frac{dm_k^j}{dt}(a_k^j)\end{bmatrix} + {}^j_k\mathbf{K}_{a^j_k;a^j_{k-1}}\, {}^j_k\mathbf{K}^{-1}_{a^j_{k-1};a^j_{k-1}} \begin{bmatrix}\bar{m}^j(a_{k-1}^j) - m_k^j(a_{k-1}^j)\\ \bar{m}^{j\prime}(a_{k-1}^j) - \frac{dm_k^j}{dt}(a_{k-1}^j)\end{bmatrix}.}$$

String times: As for non-boundary times $t \in]a^j_{k-1}, a^j_k[$, conditional on $\left(z^j_{a_{k-1}}, z^{j\prime}_{a_{k-1}}\right)$ and $\left(z^j_{a_k}, z^{j\prime}_{a_k}\right)$, $\left(z^j_t, z^{j\prime}_t\right)$ is Gaussian with mean

$$\begin{bmatrix}m_k^j(t)\\ \frac{dm_k^j}{dt}(t)\end{bmatrix} + {}^j_k\mathbf{K}_{t;\left(a^j_{k-1},a^j_k\right)}\, {}^j_k\mathbf{K}^{-1}_{\left(a^j_{k-1},a^j_k\right);\left(a^j_{k-1},a^j_k\right)} \begin{bmatrix}z^j_{a^j_{k-1}} - m_k^j(a_{k-1}^j)\\ z^{j\prime}_{a^j_{k-1}} - \frac{dm_k^j}{dt}(a_{k-1}^j)\\ z^j_{a^j_k} - m_k^j(a_k^j)\\ z^{j\prime}_{a^j_k} - \frac{dm_k^j}{dt}(a_k^j)\end{bmatrix},$$

with

$${}^j_k\mathbf{K}_{\left(a^j_{k-1},a^j_k\right);\left(a^j_{k-1},a^j_k\right)} = \begin{bmatrix}{}^j_k\mathbf{K}_{a^j_{k-1};a^j_{k-1}} & {}^j_k\mathbf{K}_{a^j_{k-1};a^j_k}\\ {}^j_k\mathbf{K}_{a^j_k;a^j_{k-1}} & {}^j_k\mathbf{K}_{a^j_k;a^j_k}\end{bmatrix}$$

and

$${}^j_k\mathbf{K}_{t;(a^j_{k-1},a^j_k)} = \begin{bmatrix}{}^j_k\mathbf{K}_{t;a^j_{k-1}} & {}^j_k\mathbf{K}_{t;a^j_k}\end{bmatrix}.$$

Hence, using once again the law of total expectation, it follows that for any $t \in]a_{k-1}^j, a_k^j[$,

$$\begin{bmatrix} \bar{m}^j(t) \\ \bar{m}^{j\prime}(t) \end{bmatrix} = \begin{bmatrix} m_k^j(t) \\ \frac{dm_k^j}{dt}(t) \end{bmatrix} + {}_k^j\mathbf{K}_{t;(a_{k-1}^j,a_k^j)} \, {}_k^j\mathbf{K}^{-1}_{(a_{k-1}^j,a_k^j);(a_{k-1}^j,a_k^j)} \begin{bmatrix} \bar{m}^j(a_{k-1}^j) - m_k^j(a_{k-1}^j) \\ \bar{m}^{j\prime}(a_{k-1}^j) - \frac{dm_k^j}{dt}(a_{k-1}^j) \\ \bar{m}^j(a_k^j) - m_k^j(a_k^j) \\ \bar{m}^{j\prime}(a_k^j) - \frac{dm_k^j}{dt}(a_k^j) \end{bmatrix}.$$

We note in particular that when $\forall\, j,k,\ m_k^j = 0$, it follows that $\bar{m}^j = 0, \bar{m}^f = 0, \bar{m}^{\nabla f} = 0$.

Global String GP Covariance Functions

As for the evaluation of ${}^j\bar{\mathbf{K}}_{u,v}$, we start by noting that the covariance function of a univariate *string GP* is the same as that of another *string GP* whose strings have the same unconditional kernels but unconditional mean functions $m_k^j = 0$, so that to evaluate univariate *string GP* kernels we may assume that $\forall\, j,k,\ m_k^j = 0$ without loss of generality. We start with the case where u and v are both boundary times, after which we will generalise to other times.

Boundary times: As previously discussed, the restriction $\left(z_t^j, z_t^{j\prime}\right)_{t\in[a_0^j,a_1^j]}$ is the *derivative Gaussian process* with mean 0 and covariance function k_1^j. Thus,

$${}^j\bar{\mathbf{K}}_{a_0^j;a_0^j} = {}_1^j\mathbf{K}_{a_0^j;a_0^j}, \qquad {}^j\bar{\mathbf{K}}_{a_1^j;a_1^j} = {}_1^j\mathbf{K}_{a_1^j;a_1^j}, \qquad {}^j\bar{\mathbf{K}}_{a_0^j;a_1^j} = {}_1^j\mathbf{K}_{a_0^j;a_1^j}. \tag{C.39}$$

We recall that conditional on the boundary conditions at or prior to a_{k-1}^j, $\left(z_{a_k^j}^j, z_{a_k^j}^{j\prime}\right)$ is Gaussian with mean

$${}_k^bM \begin{bmatrix} z_{a_{k-1}^j}^j \\ z_{a_{k-1}^j}^{j\prime} \end{bmatrix} \qquad \text{with} \qquad {}_k^bM = {}_k^j\mathbf{K}_{a_k^j;a_{k-1}^j} \, {}_k^j\mathbf{K}^{-1}_{a_{k-1}^j;a_{k-1}^j},$$

and covariance matrix

$${}_k^b\Sigma = {}_k^j\mathbf{K}_{a_k^j;a_k^j} - {}_k^bM \, {}_k^j\mathbf{K}_{a_{k-1}^j;a_k^j}.$$

Hence using Lemma C.4 with $M = \begin{bmatrix} {}_k^bM & 0 & \dots & 0 \end{bmatrix}$ where there are $(k-1)$ null block 2×2 matrices, and noting that $\left(z_{a_0^j}^j, z_{a_0^j}^{j\prime}, \dots, z_{a_{k-1}^j}^j, z_{a_{k-1}^j}^{j\prime}\right)$ is jointly Gaussian, it follows that the vector $\left(z_{a_0^j}^j, z_{a_0^j}^{j\prime}, \dots, z_{a_k^j}^j, z_{a_k^j}^{j\prime}\right)$ is jointly Gaussian, that $\left(z_{a_k^j}^j, z_{a_k^j}^{j\prime}\right)$ has

covariance matrix

$$\boxed{{}^{j}\bar{\mathbf{K}}_{a_k^j;a_k^j} = {}_{k}^{b}\Sigma + {}_{k}^{b}M\ {}^{j}\bar{\mathbf{K}}_{a_{k-1}^j;a_{k-1}^j}\ {}_{k}^{b}M^T,}$$

and that the covariance matrix between the boundary conditions at a_k^j and at any earlier boundary time $a_l^j,\ l < k$ reads:

$$\boxed{{}^{j}\bar{\mathbf{K}}_{a_k^j;a_l^j} = {}_{k}^{b}M\ {}^{j}\bar{\mathbf{K}}_{a_{k-1}^j;a_l^j}.}$$

String times: Let $u \in [a_{p-1}^j, a_p^j],\ v \in [a_{q-1}^j, a_q^j]$. By the law of total expectation, we have that

$${}^{j}\bar{\mathbf{K}}_{u;v} := \mathrm{E}\left(\begin{bmatrix} z_u^j \\ z_u^{j\prime} \end{bmatrix}\begin{bmatrix} z_v^j & z_v^{j\prime} \end{bmatrix}\right) = \mathrm{E}\left(\mathrm{E}\left(\begin{bmatrix} z_u^j \\ z_u^{j\prime} \end{bmatrix}\begin{bmatrix} z_v^j & z_v^{j\prime} \end{bmatrix}\middle|\mathcal{B}(p,q)\right)\right),$$

where $\mathcal{B}(p,q)$ refers to the boundary condtions at the boundaries of the p-th and q-th strings, in other words $\left\{z_x^j, z_x^{j\prime},\ x \in \left\{a_{p-1}^j, a_p^j, a_{q-1}^j, a_q^j\right\}\right\}$. Furthermore, using the definition of the covariance matrix under the conditional law, it follows that

$$\mathrm{E}\left(\begin{bmatrix} z_u^j \\ z_u^{j\prime} \end{bmatrix}\begin{bmatrix} z_v^j & z_v^{j\prime} \end{bmatrix}\middle|\mathcal{B}(p,q)\right) = {}_{c}^{j}\bar{\mathbf{K}}_{u;v} + \mathrm{E}\left(\begin{bmatrix} z_u^j \\ z_u^{j\prime} \end{bmatrix}\middle|\mathcal{B}(p,q)\right)\mathrm{E}\left(\begin{bmatrix} z_v^j & z_v^{j\prime} \end{bmatrix}\middle|\mathcal{B}(p,q)\right), \quad \text{(C.40)}$$

where ${}_{c}^{j}\bar{\mathbf{K}}_{u;v}$ refers to the covariance matrix between $(z_u^j, z_u^{j\prime})$ and $(z_v^j, z_v^{j\prime})$ conditional on the boundary conditions $\mathcal{B}(p,q)$, and can be easily evaluated from Theorem 4.2. In particular,

$$\text{if } p \neq q,\ {}_{c}^{j}\bar{\mathbf{K}}_{u;v} = 0,\ \text{ and if } p = q,\ {}_{c}^{j}\bar{\mathbf{K}}_{u;v} = {}_{p}^{j}\mathbf{K}_{u;v} - {}_{p}^{j}\Lambda_u \begin{bmatrix} {}_{p}^{j}\mathbf{K}_{v;a_{p-1}^j}^T \\ {}_{p}^{j}\mathbf{K}_{v;a_p^j}^T \end{bmatrix}, \quad \text{(C.41)}$$

where

$$\forall\ x, l,\ {}_{l}^{j}\Lambda_x = \begin{bmatrix} {}_{l}^{j}\mathbf{K}_{x;a_{l-1}^j} & {}_{l}^{j}\mathbf{K}_{x;a_l^j} \end{bmatrix}\begin{bmatrix} {}_{l}^{j}\mathbf{K}_{a_{l-1}^j;a_{l-1}^j} & {}_{l}^{j}\mathbf{K}_{a_{l-1}^j;a_l^j} \\ {}_{l}^{j}\mathbf{K}_{a_l^j;a_{l-1}^j} & {}_{l}^{j}\mathbf{K}_{a_l^j;a_l^j} \end{bmatrix}^{-1}.$$

We also note that

$$\mathrm{E}\left(\begin{bmatrix} z_u^j \\ z_u^{j\prime} \end{bmatrix} \middle| \mathcal{B}(p,q) \right) = {}_p^j\Lambda_u \begin{bmatrix} z^j_{a^j_{p-1}} \\ z^{j\prime}_{a^j_{p-1}} \\ z^j_{a^j_p} \\ z^{j\prime}_{a^j_p} \end{bmatrix} \text{ and } \mathrm{E}\left(\begin{bmatrix} z_v^j & z_v^{j\prime} \end{bmatrix} \middle| \mathcal{B}(p,q) \right) = \begin{bmatrix} z^j_{a^j_{q-1}} & z^{j\prime}_{a^j_{q-1}} & z^j_{a^j_q} & z^{j\prime}_{a^j_q} \end{bmatrix} {}_q^j\Lambda_v^T.$$

Hence, taking the expectation with respect to the boundary conditions on both sides of Equation (C.40), we obtain:

$$\forall\ u \in [a^j_{p-1}, a^j_p],\ v \in [a^j_{q-1}, a^j_q],\quad {}^j\bar{\mathbf{K}}_{u;v} = {}_c^j\bar{\mathbf{K}}_{u;v} + {}_p^j\Lambda_u \begin{bmatrix} {}^j\bar{\mathbf{K}}_{a^j_{p-1};a^j_{q-1}} & {}^j\bar{\mathbf{K}}_{a^j_{p-1};a^j_q} \\ {}^j\bar{\mathbf{K}}_{a^j_p;a^j_{q-1}} & {}^j\bar{\mathbf{K}}_{a^j_p;a^j_q} \end{bmatrix} {}_q^j\Lambda_v^T,$$

where ${}_c^j\bar{\mathbf{K}}_{u;v}$ is provided in Equation (C.41).

Appendix D

Derivations of Chapter 7

D.1 GP Models and General-Purposeness

Let us consider a Bayesian model aiming at infering a latent function f on which a Gaussian process prior with covariance function k is placed. Let $p\left(\mathcal{D}|\mathbf{f}\right)$ denote the likelihood model, where $\mathbf{f}$ is the vector of values of f at some training inputs, and $\mathbf{f}^*$ corresponds to the values of f at some test inputs. If $p\left(\mathcal{D}|\mathbf{f}\right)$ is a continuous and bounded function of $\mathbf{f}$, and k_{θ_i} is a sequence of kernels converging to k pointwise, then the posterior probability density functions $p\left(\mathbf{f}^*|\mathcal{D}, k_{\theta_i}\right)$ converge to $p\left(\mathbf{f}^*|\mathcal{D}, k\right)$, and consequently the conditional random variables $\mathbf{f}^*|\mathcal{D}, k_{\theta_i}$ converge to $\mathbf{f}^*|\mathcal{D}, k$ in distribution. Moreover, the nodel evidences $p(\mathcal{D}|k_{\theta_i})$ converge to $p(\mathcal{D}|k)$.

In effect, by Baye's theorem

$$p\left(\mathbf{f}^*|\mathcal{D}, k_{\theta_i}\right) = \frac{p\left(\mathbf{f}^*|k_{\theta_i}\right) \int p\left(\mathcal{D}|\mathbf{f}\right) p\left(\mathbf{f}|\mathbf{f}^*, k_{\theta_i}\right) d\mathbf{f}}{\int p\left(\mathcal{D}|\mathbf{f}\right) p\left(\mathbf{f}|k_{\theta_i}\right) d\mathbf{f}}. \tag{D.1}$$

As the (Gaussian) conditional density $p\left(\mathbf{f}|\mathbf{f}^*, k\right)$ only depends on the kernel k as a continuous function of the Gram matrices $\mathbf{K}_{\mathbf{x}^*,\mathbf{x}^*}$ and $\mathbf{K}_{\mathbf{x},\mathbf{x}^*}$, pointwise convergence of the kernels k_{θ_i} to k implies convergence of the densities $p\left(\mathbf{f}|\mathbf{f}^*, k_{\theta_i}\right)$ to $p\left(\mathbf{f}|\mathbf{f}^*, k\right)$, and consequently weak convergence of the probability measures $p\left(\mathbf{f}|\mathbf{f}^*, k_{\theta_i}\right) d\mathbf{f}$ to $p\left(\mathbf{f}|\mathbf{f}^*, k\right) d\mathbf{f}$. As $p\left(\mathcal{D}|\mathbf{f}\right)$ is assumed continuous and bounded, by definition of weak convergence of measures,

$$\int p\left(\mathcal{D}|\mathbf{f}\right) p\left(\mathbf{f}|\mathbf{f}^*, k_{\theta_i}\right) d\mathbf{f} \underset{i\to+\infty}{\longrightarrow} \int p\left(\mathcal{D}|\mathbf{f}\right) p\left(\mathbf{f}|\mathbf{f}^*, k\right) d\mathbf{f}.$$

Finally, using a similar reasoning we obtain

$$\int p\left(\mathcal{D}|\mathbf{f}\right) p\left(\mathbf{f}|k_{\theta_i}\right) d\mathbf{f} \underset{i\to+\infty}{\longrightarrow} \int p\left(\mathcal{D}|\mathbf{f}\right) p\left(\mathbf{f}|k\right) d\mathbf{f}, \tag{D.2}$$

and

$$p\left(\mathbf{f}^*|k_{\theta_i}\right) \underset{i\to+\infty}{\longrightarrow} p\left(\mathbf{f}^*|k\right).$$

Hence, using Equation (D.1) we conclude that $p\left(\mathbf{f}^*|\mathcal{D}, k_{\theta_i}\right)$ converges to $p\left(\mathbf{f}^*|\mathcal{D}, k\right)$. Note that Equation (D.2) establishes the convergence of the model evidences $p(\mathcal{D}|k_{\theta_i})$ to $p(\mathcal{D}|k)$.

D.2 Differentiability of Stationary GSKs

Let us prove the following proposition.

Proposition: A mean zero stationary Gaussian process with *stationary generalized spectral kernel* as covariance function is p times continuously differentiable in the mean square sense if and only if a mean zero stationary Gaussian process with covariance function h is.

Proof p times differentiability of a stationary GP in the mean square sense is equivalent to $2p$ times differentiability of its covariance function at 0 (see Adler and Taylor, 2011, Ch. 2). It is easy to see that if h is $2p$ times differentiable at 0, then so will the corresponding stationary generalized spectral kernel. Reciprocally, a simple reasoning by contradiction allows us to conclude that if h is not at least $2p$ times differentiable at 0, $h(\tau \odot \gamma_k)\cos(2\pi\omega_k^T\tau)$ and subsequently the corresponding stationary generalized spectral kernel cannot be. ■

D.3 Integrability of Stationary GSKs

Let us prove the following proposition.

Proposition: A stationary generalized spectral kernel is integrable if and only if $\forall k \leq K$, either $\gamma_k \neq 0$ or $\alpha_k = 0$.

Proof If for any k we have $\gamma_k = 0$ and $\alpha_k > 0$, then the term $\alpha_k h(0)\cos(2\pi\omega_k^T\tau)$ and consequently k_{SGS} will not vanish, whence k_{SGS} will not be integrable. On the other hand when $\forall k \leq K$, either $\gamma_k \neq 0$ or $\alpha_k = 0$, as $|k_{\text{SGS}}(\tau; K)| \leq \sum_{k=1}^{K} \alpha_k |h(\tau \odot \gamma_k)|$, the integrability of h allows us to conclude. ■

D.4 Construction of General-Purpose Kernels

In this section we provide a detailed derivation of *generalized spectral kernels*. In particular, we prove that they are general-purpose (i.e. pointwise dense in the family of all real-valued continuous bounded kernels). We begin by stating and proving a few lemmas that we will later use.

Lemma D.1 *The family of* ***discrete*** *finite* ***symmetric*** *measures on* $(\mathbb{R}^d \times \mathbb{R}^d, \mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d))$ *is weakly dense in the space of bimeasures satisfying the conditions of Definition 7.6.*

Proof Firstly, we recall that bimeasures satisfying the conditions of Definition 7.6 are necessarily symmetric and positive semi-definite — that is they satisfy

$$\forall A_i, A_j \in \mathcal{B}(\mathbb{R}^d), \quad \mu_F(A_i, A_j) = \mu_F(A_j, A_i) \tag{D.3}$$

and

$$\forall A_i \in \mathcal{B}(\mathbb{R}^d),\ c_i \in \mathbb{C}, \quad \sum_{i=1}^{n}\sum_{j=1}^{n} c_i \mu_F(A_i, A_j)\bar{c}_j \geq 0. \tag{D.4}$$

Secondly, as noted in (Dehay and Moché, 1992, §1.1), positive semi-definite bimeasures defined on $\mathcal{B}(\mathbb{R}^d) \times \mathcal{B}(\mathbb{R}^d)$ are fully fledged measures on $\mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d)$. We note however that in general, unlike in the stationary case, they are signed measures (i.e. we may have $\mu_F(A_i, A_j) < 0$ when $A_i \neq A_j$).

Thirdly, Definition 7.6 requires μ_F to have finite total variation, which implies that μ_F is finite given that

$$|\mu_F(\mathbb{R}^d, \mathbb{R}^d)| \leq |\mu_F|(\mathbb{R}^d, \mathbb{R}^d) < \infty. \tag{D.5}$$

Fourthly, as the family of finite *discrete* measures on $\mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d)$ is weakly dense in the family of all finite measures on $\mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d)$ (see Fabec, 2000, corollary I.10), it is also weakly dense in the family of all bimeasures satisfying the conditions of Definition 7.6.

Lastly, if we consider μ_F a bimeasure satisfying the conditions of Definition 7.6, and $\hat{\mu}_n$ a sequence of finite discrete measures on $\mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d)$ that converges to μ_F weakly, then the sequence of finite discrete measures

$$\mu_n^s(A, B) := \frac{1}{2}\left(\mu_n(A, B) + \mu_n(B, A)\right) \tag{D.6}$$

are symmetric, and by symmetry of μ_F they also converge weakly to μ_F. This concludes the proof . ■

Lemma D.2 *The family of discrete finite symmetric measures on* $(\mathbb{R}^d \times \mathbb{R}^d, \mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d))$ *that* ***satisfy the condition****,*

$$\forall i \in \mathbb{N}^*,\ \omega_i \in \mathbb{R}^d, \quad \#\left\{j,\ s.t.\ \omega_j \neq \omega_i \text{ and } \mu\left(\{\omega_i\}, \{\omega_j\}\right) \neq 0\right\} \leq 1 \tag{D.7}$$

is weakly dense in the space of bimeasures satisfying the conditions of Definition 7.6.

Proof The only difference with Lemma D.1 is the requirement that the approximating measures should satisfy condition (D.7). Hence, all we need to prove is that condition (D.7) does not affect the weak density property. A sufficient condition for this to hold true is that for any discrete finite symmetric measure $\mu_{\text{sing.}}$, we may always find a sequence of discrete finite symmetric measures satisfying condition (D.7) and that converge to $\mu_{\text{sing.}}$ weakly.[1] This indeed holds true, which we prove below.

By definition, any discrete finite symmetric (signed) measure on $\mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d)$ takes the form

$$\mu_{\text{sing.}}(A, B) = \sum_{i=1}^{+\infty}\sum_{j=1}^{+\infty} \mu_{ij}\delta_{\{\omega_i\}}(A)\delta_{\{\omega_j\}}(B) \tag{D.8}$$

where $\delta_{\{x\}}$ is the Dirac measure with support the singleton $\{x\}$, $\mu_{ij} = \mu_{ji} \in \mathbb{R}$ and $\sum_{i=1}^{+\infty}\sum_{j=1}^{+\infty}\mu_{ij} < +\infty$. Moreover, we may always find a sequence of small 'jitters' $\epsilon_n^i \in \mathbb{R}^d$ such that

$$\hat{\mu}_n(A, B) := \sum_{i=1}^{+\infty}\sum_{j=1}^{+\infty} \mu_{ij}\delta_{\{\epsilon_n^i + \omega_i\}}(A)\delta_{\left\{\epsilon_n^j + \omega_j\right\}}(B) \tag{D.9}$$

satisfies condition (D.7) and such that

$$\lim_{n \to +\infty} \sup_i ||\epsilon_n^i|| = 0. \tag{D.10}$$

[1] A dense in B dense in C implies A dense in C.

To see why, let $u_n^i \in \mathbb{R}^d$ be a sequence of elements that are distinct across a section i, and that converges uniformly to 0. For a given n, sequentially define ϵ_n^i to strictly lie between the ball $B(0, ||u_n^i||)$ and the ball $B(0, ||u_{n+1}^i||)$ and such that

$$\forall l \leq i, \quad \#\left\{j \leq i, \text{ s.t. } \omega_l + \epsilon_n^l \neq \omega_j + \epsilon_n^j \text{ and } \mu(\{\omega_l + \epsilon_n^l\}, \{\omega_j + \epsilon_n^j\}) \neq 0\right\} \leq 1;$$

in other words, such that condition (D.7) is satisfied up to order i. This is always possible because, condition (D.7) is always satisfied for $i = 1$ and when it is is satisfied up to order $i-1$, there are at most a countable number of values for ϵ_n^i that can prevent condition (D.7) to hold true up to order i — indeed these values ought to be such that $\omega_i + \epsilon_n^i$ lies in the union of the supports of the $i-1$ measures $A \to \mu\left(\{\omega_l + \epsilon_n^l\}, A\right)$ with $l \leq i-1$, which is at most countable. As we have assumed that the elements u_n^i are distinct for a given i, there will be an uncountable number of elements ϵ_n^i between the two balls $B(0, ||u_n^i||)$ and $B(0, ||u_{n+1}^i||)$. Moreover, given that ϵ_n^i strictly lies between the ball $B(0, ||u_n^i||)$ and the ball $B(0, ||u_{n+1}^i||)$, the uniform convergence of u_n^i to zero implies the uniform convergence of e_n^i to zero (i.e. Equation (D.10) holds), and for every n, condition (D.7) is satisfied by the resulting $\hat{\mu}_n$.

Furthermore, $\hat{\mu}_n$ is easily found be discrete, finite and symmetric from first principles. Finally, for every continuous bounded function f, as n goes to $+\infty$,

$$\int f d\hat{\mu}_n := \sum_{i=1}^{+\infty} \sum_{j=1}^{+\infty} \mu_{ij} f(\omega_i + \epsilon_n^i, \omega_j + \epsilon_n^j) \tag{D.11}$$

converges to

$$\int f d\mu_{\text{sing.}} := \sum_{i=1}^{+\infty} \sum_{j=1}^{+\infty} \mu_{ij} f(\omega_i, \omega_j) \tag{D.12}$$

(i.e. by definition μ_n weakly converges to $\mu_{\text{sing.}}$). Indeed, we may exchange limit and sum by dominated convergence theorem after noting that $f(\omega_i + \epsilon_n^i, \omega_j + \epsilon_n^j)$ is uniformly bounded by M where M is an upper bound of f, which we recall is assumed bounded, and $\mu(\mathbb{R}^d \times \mathbb{R}^d) < +\infty$. We may then use the continuity of f to conclude.

This ends the proof of Lemma D.2. ■

Lemma D.3 *Any discrete measure μ on $(\mathbb{R}^d \times \mathbb{R}^d, \mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d))$ that satisfies the condition (D.7) can be decomposed as*

$$\mu(A, B) = \sum_{k=1}^{+\infty} \mu_k^{sq}(A, B), \tag{D.13}$$

where μ_k^{sq} are measures supported on disjoint sets of the form $\{\omega_{2k}, \omega_{2k+1}\} \times \{\omega_{2k}, \omega_{2k+1}\}$, with $\omega_{2k}, \omega_{2k+1} \in \mathbb{R}^d$.

Proof Let μ be a discrete measure on $\mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d)$ that satisfies the condition (D.7):

$$\mu(A, B) = \sum_{i=1}^{+\infty} \sum_{j=1}^{+\infty} \mu_{ij} \delta_{\{\omega_i\}}(A) \delta_{\{\omega_j\}}(B). \tag{D.14}$$

We may rearrange its marginal support $\{\dots, \omega_i, \dots\}$ such that

$$\begin{aligned} \forall k \neq l, \quad \mu(\{\omega_{2l+1}\}, \{\omega_{2k}\}) &= \mu(\{\omega_{2l}\}, \{\omega_{2k}\}) \\ &= \mu(\{\omega_{2l+1}\}, \{\omega_{2k+1}\}) \\ &= 0, \end{aligned} \tag{D.15}$$

we may have

$$\mu(\{\omega_{2k+1}\}, \{\omega_{2k}\}) \neq 0,$$

and by convention, when i is such that

$$\forall j \neq i, \quad \mu(\{\omega_i\}, \{\omega_j\}) = 0,$$

we choose to associate it to an even index $2k$ and we add a new 'fake' singleton to the support, which we denote $\{\omega_{2k+1}\}$. ω_{2k+1} is fake in the sense that $\forall j \in \mathbb{N}^*, \mu(\{\omega_{2k+1}\}, \{\omega_j\}) = 0$, i.e. it does not change the measure μ. With this convention in mind, we may decompose μ as

$$\begin{aligned} \mu(A, B) &= \mu\left(A \cap \cup_k \{\omega_{2k}, \omega_{2k+1}\}, B \cap \cup_l \{\omega_{2l}, \omega_{2l+1}\}\right) \\ &= \mu\left(\cup_k A \cap \{\omega_{2k}, \omega_{2k+1}\}, \cup_l B \cap \{\omega_{2l}, \omega_{2l+1}\}\right) \\ &= \sum_k^{+\infty} \sum_l^{+\infty} \mu\left(A \cap \{\omega_{2k}, \omega_{2k+1}\}, B \cap \{\omega_{2l}, \omega_{2l+1}\}\right) \\ &= \sum_k^{+\infty} \mu\left(A \cap \{\omega_{2k}, \omega_{2k+1}\}, B \cap \{\omega_{2k}, \omega_{2k+1}\}\right), \end{aligned}$$

where the last equality results from Equation (D.15). We then conclude by defining

$$\mu_k^{\text{sq}}(A, B) := \mu\left(A \cap \{\omega_{2k}, \omega_{2k+1}\}, B \cap \{\omega_{2k}, \omega_{2k+1}\}\right). \tag{D.16}$$

■

Lemma D.4 *Any discrete measure μ on $(\mathbb{R}^d \times \mathbb{R}^d, \mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d))$ that can be decomposed as*

$$\mu(A, B) = \sum_{k=1}^{+\infty} \mu_k^{sq}(A, B), \tag{D.17}$$

where μ_k^{sq} are measures supported on disjoint sets of the form $\{\omega_{2k}, \omega_{2k+1}\} \times \{\omega_{2k}, \omega_{2k+1}\}$, with $\omega_{2k}, \omega_{2k+1} \in \mathbb{R}^d$ is positive definite if and only if for every k, the 2×2 matrix

$$\Sigma_k = [\mu_k^{sq}(\{\omega_{2k+i}\}, \{\omega_{2k+j}\})]_{i,j \in \{0,1\}} \tag{D.18}$$

is positive semi-definite.

Proof As μ_k^{sq} is supported on $\{\omega_{2k}, \omega_{2k+1}\} \times \{\omega_{2k}, \omega_{2k+1}\}$, μ_k^{sq} is positive definite if and only if

$$\forall c_0, c_1 \in \mathbb{C}, \quad \sum_{i=0}^{1} \sum_{j=0}^{1} c_i \mu_k^{\text{sq}}(\{\omega_{2k+i}\}, \{\omega_{2k+j}\}) \bar{c}_j \geq 0, \tag{D.19}$$

which is equivalent to the positive semi-definitiveness of Σ_k. Hence, when Σ_k are all positive semi-definite, μ is positive definite too as sum of such measures. Reciprocally, when μ is positive definite,

$$\begin{aligned} \forall c_0, c_1 \in \mathbb{C}, \quad & \sum_{i=0}^{1} \sum_{j=0}^{1} c_i \mu_k^{\text{sq}}(\{\omega_{2k+i}\}, \{\omega_{2k+j}\}) \bar{c}_j \\ & = \sum_{i=0}^{1} \sum_{j=0}^{1} c_i \mu(\{\omega_{2k+i}\}, \{\omega_{2k+j}\}) \bar{c}_j \\ & \geq 0, \end{aligned}$$

and consequently Σ_k is positive semi-definite. ■

Lemma D.5 *The family of discrete finite symmetric measures on $(\mathbb{R}^d \times \mathbb{R}^d, \mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d))$ that satisfy the condition (D.7) and are **positive definite** is weakly dense in the space of bimeasures satisfying the conditions of Definition 7.6.*

The difference between this lemma and Lemma D.2 is the positive definitiveness requirement; we simply need to prove that this requirement does not alter the weak density property established in Lemma D.2. Intuitively, as bimeasures satisfying the conditions of Definition 7.6 are positive definite measures this comes as no surprise.

Proof Before we begin, we recall from Lemma D.3 that if

$$\mu(A,B) = \sum_{i=1}^{+\infty}\sum_{j=1}^{+\infty} \mu_{ij}\delta_{\{\omega_i\}}(A)\delta_{\{\omega_j\}}(B) \tag{D.20}$$

is a discrete finite symmetric measures on $\mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d)$ that satisfies the condition (D.7), then we may decompose μ as

$$\mu(A,B) = \sum_{k=1}^{+\infty} \mu_k^{\text{sq}}(A,B), \tag{D.21}$$

where μ_k^{sq} is the measure supported on $\{\omega_{2k},\omega_{2k+1}\} \times \{\omega_{2k},\omega_{2k+1}\}$, and such that

$$\Sigma_k = [\mu_k^{\text{sq}}(\{\omega_{2k+i}\},\{\omega_{2k+j}\})]_{i,j\in\{0,1\}}\,. \tag{D.22}$$

Let μ_F be a bimeasure satisfying the conditions of Definition 7.6 — in particular μ_F is positive definite. Moreover, it follows from Lemma D.2 that there exists a sequence μ_n of discrete finite symmetric measures, which we may decompose as in Equation (D.21)

$$\mu_n(A,B) = \sum_{k=1}^{+\infty} \mu_{k,n}^{\text{sq}}(A,B), \tag{D.23}$$

that converge weakly to μ_F. Let us write $\Sigma_{k,n}$ the Gram matrix corresponding to the measure $\mu_{k,n}^{\text{sq}}$ as in Equation (D.22). Without loss of generality, we may choose $\mu_{k,n}^{\text{sq}}$ to have the same support $\{\omega_{2k},\omega_{2k+1}\} \times \{\omega_{2k},\omega_{2k+1}\}$ for every n. Indeed, we may always choose $\{\dots,\omega_i,\dots\} = \bigcup_n \{\dots,\omega_{k,n},\dots\}$,[2] which is again a countable set as countable union of countable sets.

Now, let us consider the sequence of measures

$$\hat{\mu}_n(A,B) = \sum_{k=1}^{+\infty} \hat{\mu}_{k,n}^{\text{sq}}(A,B), \tag{D.24}$$

where $\hat{\mu}_{k,n}^{\text{s.q.}}$ only differs from $\mu_{k,n}^{\text{s.q.}}$ through its 2×2 Gram matrix $\hat{\Sigma}_{k,n}$, which we choose to be the closest symmetric positive semi-definite matrix to $\Sigma_{k,n}$ (in Frobenius norm). $\hat{\mu}_n$ is discrete, finite, symmetric, but also positive definite as a result of Lemma D.4. Using the fact that μ_n weakly converges to μ_F, and that μ_F is positive definite, we get that $||\hat{\Sigma}_{k,n} - \Sigma_{k,n}||$ converges uniformly to 0. Consequently, for every continuous bounded function f, $\int f d\mu_{k,n}^{\text{sq}} - \int f d\hat{\mu}_{k,n}^{\text{sq}}$ convergence uniformly to zero, and therefore

[2] Letting some Σ_k be 0 if necessary.

we may exchange limit in n and sum over k to get that $\int f d\mu_n - \int f d\hat{\mu}_n$ converges to 0. This proves that $\int f d\hat{\mu}_n$ converges to $\int f d\mu_F$ as well. ■

Lemma D.6 *A 2×2 matrix Σ is positive semi-definite if and only if there exists an upper triangular matrix U such that $\Sigma = U^T U$.*

Proof If U is upper triangular, then $\Sigma := U^T U$ is easily found to be positive semi-definite. Reciprocally, when

$$\Sigma = \begin{pmatrix} a & b \\ b & c \end{pmatrix}, \tag{D.25}$$

is a positive semi-definite matrix, $a \geq 0$ and $b^2 \leq ac$, and we may conclude by taking

$$U = \begin{pmatrix} \sqrt{a} & \frac{b}{\sqrt{a}} \\ 0 & \sqrt{c - \frac{b^2}{a}} \end{pmatrix}, \tag{D.26}$$

or

$$U = -\begin{pmatrix} \sqrt{a} & \frac{b}{\sqrt{a}} \\ 0 & \sqrt{c - \frac{b^2}{a}} \end{pmatrix}. \tag{D.27}$$

Note that we need not require U to have positive entries or even positive diagonal elements. ■

Summary: Overall, combining Lemmas D.5 and D.6, we have that discrete, finite, symmetric, and positive definite measures on $(\mathbb{R}^d \times \mathbb{R}^d, \mathcal{B}(\mathbb{R}^d \times \mathbb{R}^d))$ that are of the form

$$\mu_{\text{sing.}}(A, B) = \sum_{k=1}^{+\infty} \mu_k^{\text{sq}}(A, B), \tag{D.28}$$

where μ_k^{sq} is the measure supported on $\{\omega_{2k}, \omega_{2k+1}\} \times \{\omega_{2k}, \omega_{2k+1}\}$, for distinct $\omega_i \in \mathbb{R}^d$, and such that

$$[\mu_k^{\text{sq}}(\{\omega_{2k+i}\}, \{\omega_{2k+j}\})]_{i,j \in \{0,1\}} = U_k^T U_k, \tag{D.29}$$

for any upper triangular 2×2 matrix U_k, are weakly dense in the space of all bimeasures satisfying the conditions of Definition 7.6. Consequently, the resulting kernels

$$\begin{aligned} k_{\text{sing.}}(x, y) &:= \int e^{2\pi i(\omega_1^T x - \omega_2^T y)} d\mu_{\text{sing.}}(d\omega_1, d\omega_2) \\ &= \sum_{k=1}^{+\infty} \Upsilon_k(x)^* \Upsilon_k(y) \end{aligned} \tag{D.30}$$

where $\Upsilon_k(x) := U_k \begin{pmatrix} e^{2\pi i x^T \omega_k^1} \\ e^{2\pi i x^T \omega_k^2} \end{pmatrix}$, $\omega_k^1 := \omega_{2k}$, and $\omega_k^2 := \omega_{2k+1}$ are pointwise dense in the space all complex-valued strongly harmonizable kernels, which implies it is also dense in the family of all complex-valued continuous bounded kernels (see Corollary 7.8).

The pointwise density property is obviously preserved when we truncate the above series by taking the first K terms; this leads to the trigonometric kernel k_{CNS} of Equation (7.12). As a result, the family of real-parts $\mathcal{R}e(k_{\text{CNS}})$ is pointwise dense in the family of all real-valued continuous bounded kernels. Finally, the pointwise density of generalized spectral kernels results from the fact that $\mathcal{R}e(k_{\text{CNS}})$ is recovered as the subfamily corresponding to $\rho = \rho^*$.

Appendix E

Derivations of Chapter 8

E.1 Extended RJ-MCMC

Proposition E.1 *Let* $(x_k)_{k\in\mathbb{N}} \sim \pi_X$, $x_k \in \mathcal{X} \subseteq \mathbb{R}^d$, *where* π_X *is some stochastic process. Let* $(\beta_k)_{k\in\mathbb{N}}$, $\beta_k \in \mathcal{B} \subseteq \mathbb{R}^p$ *be i.i.d. distributed as per some distribution admitting a pdf* p *and that is independent of* π_X, *and let* $\hat{p}$ *be a pdf with the same support as* p. *Let* $K \sim \pi(K)$, $K \in \mathcal{I} := \mathbb{N} \cap [k_{\min}, +\infty)$, $k_{\min} \in \mathbb{N}$, *where* $\pi(K)$ *is a discrete probability distribution. Let* $\mathcal{C} = \bigcup_{K\in\mathcal{I}} \{K\} \times (\mathcal{X} \times \mathcal{B})^K$. *We denote* $\pi(dx_1, \dots, dx_K)$ *the joint probability measure of* $(x_1, \dots, x_K)$, *and* $\pi(dx_{K+1}|x_1, \dots, x_K)$ *the marginal conditional probability measure of* x_{K+1} *given* $x_1, \dots, x_K$. *Finally, let* $p(\mathcal{D}|x_1, \dots, x_K)$ *be the strictly positive pdf of some random variable* $\mathcal{D}$ *given* $x_1, \dots, x_K$.

We denote P *the Markov transition kernel operating on the input space* $\mathcal{C}$ *as follows:*

$$P\left((K, x_1, \beta_1, \dots, x_K, \beta_K), (K', dx'_1, d\beta'_1, \dots, dx'_{K'}, d\beta'_{K'})\right) = \tag{E.1}$$

$$\begin{cases} j_K(K')\pi(dx'_{K'}, \dots, dx'_{K+1}|x_1, \dots, x_K) \prod_{k=1}^{K} \delta_{(x_k,\beta_k)}(dx'_k, d\beta'_k) \prod_{k=K+1}^{K'} \hat{p}(\beta'_k)d\beta'_k & \text{if } K' > K \\ j_K(K') \prod_{k\in\Omega_K(K')} \delta_{(x_k,\beta_k)}(dx'_k, d\beta'_k) & \text{otherwise} \end{cases}, \tag{E.2}$$

where for every K, $j_K(.)$ *is a pmf with support* $\mathbb{N} \cap [k_{\min}, +\infty)$ *such that* $j_K(K+1) > 0$, $j_K(K) = 0$ *and for all* $K > k_{\min}$, $j_K(K-1) > 0$, $\delta_{(x,\beta)}(dx', d\beta')$ *is the Dirac measure on* $\mathcal{X} \times \mathcal{B}$ *with support* (x, β), *and* $\Omega_K(K')$ *is a set of* K' *distinct indices from* $[1 \dots K]$.

The Hastings Markov transition kernel Q operating on the input space $\mathcal{C}$, whose proposal transition kernel is P and whose acceptance probability reads

$$r_{K\to K'} = \min\left(1, \frac{p\left(\mathcal{D}|x'_1,\dots,x'_{K'}\right)}{p\left(\mathcal{D}|x_1,\dots,x_K\right)} \frac{\pi(K')}{\pi(K)} \frac{j_{K'}(K)}{j_K(K')} \prod_{k\in\mathcal{K}} \frac{p(\beta'_k)}{\hat{p}(\beta'_k)}\right), \tag{E.3}$$

when $K' > K$ and

$$r_{K\to K'} = \min\left(1, \frac{p\left(\mathcal{D}|x'_1,\dots,x'_{K'}\right)}{p\left(\mathcal{D}|x_1,\dots,x_K\right)} \frac{\pi(K')}{\pi(K)} \frac{j_{K'}(K)}{j_K(K')} \prod_{k\in\mathcal{K}} \frac{\hat{p}(\beta'_k)}{p(\beta'_k)}\right), \tag{E.4}$$

otherwise, where $\mathcal{K}$ is the set of indices that differ between the current state and the proposal, is irreducible with respect to K and satisfies the detailed-balance condition relative to the distribution $K, x_1, \beta_1, \dots, x_K, \beta_K \mid \mathcal{D}$.

Proof First of all, as $j_K(K+1) > 0$ for all K, and $j_K(K-1) > 0$ for all $K > k_{\min}$, it is easy to see that the probability of reaching a number of components K' from K in $|K'-K|$ steps is strictly positive, no matter K and K', which proves the irreducibility with respect to K. For the rest of this proof we introduce the state variables $\boldsymbol{u}_K := (x_1, \beta_1, \dots, x_K, \beta_K)$ and $\boldsymbol{u}'_{K'} := (x'_1, \beta'_1, \dots, x'_{K'}, \beta'_{K'})$ to simplify the notations.

To prove detailed-balance, let us denote ψ_K the measure $K, \boldsymbol{u}_K \mid \mathcal{D}$, where K is held fixed.[1] We need to show that for every $K, K' \in \mathcal{I}$ and for every multidimensional intervals $A_K \subseteq (\mathcal{X}\times\mathcal{B})^K$ and $A_{K'} \subseteq (\mathcal{X}\times\mathcal{B})^{K'}$,

$$\int_{A_K}\int_{A_{K'}} \psi_K(d\boldsymbol{u}_K) Q\left((K,\boldsymbol{u}_K),(K',d\boldsymbol{u}'_{K'})\right) = \int_{A_{K'}}\int_{A_K} \psi_{K'}(d\boldsymbol{u}'_{K'}) Q\left((K',\boldsymbol{u}'_{K'}),(K,d\boldsymbol{u}_K)\right). \tag{E.5}$$

By symmetry we may restrict ourselves to the case $K \leq K'$ without loss of generality. For the sake of clarity, we rewrite the acceptance probability as

$$r_{K\to K'} := r\left((K,\boldsymbol{u}_K),(K',\boldsymbol{u}'_{K'})\right),$$

so as to make the dependency in the state variables explicit. We note that we may also rewrite our transition kernel as

$$Q\left((K,\boldsymbol{u}_K),(K',A_{K'})\right) := \int_{A_{K'}} Q\left((K,\boldsymbol{u}_K),(K',d\boldsymbol{u}'_{K'})\right)$$

[1] This is not a probability measure as it does not sum to one.

$$= \int_{A_{K'}} r\left((K, \boldsymbol{u}_K), (K', \boldsymbol{u}'_{K'})\right) P\left((K, \boldsymbol{u}_K), (K', d\boldsymbol{u}'_{K'})\right) + s\left(\boldsymbol{u}_K, K\right) \mathbb{1}\left((K, \boldsymbol{u}_K) \in \{K'\} \times A_{K'}\right), \tag{E.6}$$

where

$$s\left(\boldsymbol{u}_K, K\right) := \sum_{K' \in \mathcal{I}} \int_{(\mathcal{X} \times \mathcal{B})^{K'}} \left(1 - r\left((K, \boldsymbol{u}_K), (K', \boldsymbol{u}'_{K'})\right)\right) P\left((K, \boldsymbol{u}_K), (K', d\boldsymbol{u}'_{K'})\right)$$

is the probability of not moving from the current state. Plugging this into Equation (E.5), we get that for detailed-balance to hold, we must have

$$\begin{aligned} &\int_{A_K} \int_{A_{K'}} \psi_K(d\boldsymbol{u}_K) P\left((K, \boldsymbol{u}_K), (K', d\boldsymbol{u}'_{K'})\right) r\left((K, \boldsymbol{u}_K), (K', \boldsymbol{u}'_{K'})\right) \\ &= \int_{A_{K'}} \int_{A_K} \psi_{K'}(d\boldsymbol{u}'_{K'}) P\left((K', \boldsymbol{u}'_{K'}), (K, d\boldsymbol{u}_K)\right) r\left((K', \boldsymbol{u}'_{K'}), (K, \boldsymbol{u}_K)\right). \end{aligned} \tag{E.7}$$

This obviously holds when $K = K'$ as both terms will be 0, given that every proposal move changes the number of spectral components (we recall that $j_K(K) = 0$). Let us consider the case $K < K'$. We have that

$$\begin{aligned} \psi_K(d\boldsymbol{u}_K) P\left((K, \boldsymbol{u}_K), (K', d\boldsymbol{u}'_{K'})\right) = &\frac{p(\mathcal{D}|x_1, \ldots, x_K)}{p(\mathcal{D})} \pi(K) \pi\left(dx_1, \ldots, dx_K, dx'_{K+1}, \ldots, dx'_{K'}\right) \times \\ &j_K(K') \prod_{k=1}^{K} p(\beta_k) d\beta_k \prod_{l=K+1}^{K'} \hat{p}(\beta'_l) d\beta'_l \prod_{k=1}^{K} \delta_{(x_k, \beta_k)}\left(dx'_k, d\beta'_k\right) \end{aligned} \tag{E.8}$$

and

$$\begin{aligned} \psi_{K'}(d\boldsymbol{u}'_{K'}) P\left((K', \boldsymbol{u}'_{K'}), (K, d\boldsymbol{u}_K)\right) = &\frac{p(\mathcal{D}|x'_1, \ldots, x'_{K'})}{p(\mathcal{D})} \pi(K') \pi\left(dx'_1, \ldots, dx'_{K'}\right) \times j_{K'}(K) \\ &\prod_{k=1}^{K'} p(\beta'_k) d\beta'_k \prod_{k \in \Omega_{K'}(K)} \delta_{\left(x'_k, \beta'_k\right)}\left(dx_k, d\beta_k\right), \end{aligned} \tag{E.9}$$

where

$$p(\mathcal{D}) = \sum_{K \in \mathcal{I}} \int_{(\mathcal{X} \times \mathcal{B})^K} p(\mathcal{D}|x_1, \ldots, x_K) \pi(K) \pi(dx_1, \ldots, dx_K) \prod_{k=1}^{K} p(\beta_k) d\beta_k$$

is the model evidence. Let us denote

$$U'_K = \{(x'_1, \beta'_1, \ldots, x'_K, \beta'_K) \text{ s.t. } \boldsymbol{u}'_{K'} := (x'_1, \beta'_1, \ldots, x'_{K'}, \beta'_{K'}) \in A'_K\},$$

and let us define $B_{K'}$ such that[2]

$$A_{K'} = U'_K \times B_{K'}.$$

It follows from Equation (E.8) that

$$\begin{aligned}
&\int_{A_K} \int_{A_{K'}} \psi_K(d\boldsymbol{u}_K) P\left((K, \boldsymbol{u}_K), (K', d\boldsymbol{u}'_{K'})\right) \\
&= \int_{A_K} \int_{U'_K \times B_{K'}} \frac{p(\mathcal{D}|x_1, \ldots, x_K)}{p(\mathcal{D})} \pi(K) \pi\left(dx_1, \ldots, dx_K, dx'_{K+1}, \ldots, dx'_{K'}\right) \times \\
&\qquad r\left((K, \boldsymbol{u}_K), (K', \boldsymbol{u}'_{K'})\right) j_K(K') \prod_{k=1}^{K} p(\beta_k) d\beta_k \prod_{l=K+1}^{K'} \hat{p}(\beta'_l) d\beta'_l \prod_{k=1}^{K} \delta_{(x_k, \beta_k)}(dx'_k, d\beta'_k) \\
&= \int_{A_K \cap U'_K} \int_{B_{K'}} \frac{p(\mathcal{D}|x_1, \ldots, x_K)}{p(\mathcal{D})} \pi(K) \pi\left(dx_1, \ldots, dx_K, dx'_{K+1}, \ldots, dx'_{K'}\right) \times \\
&\qquad r\left((K, \boldsymbol{u}_K), (K', \boldsymbol{u}'_{K'})\right) j_K(K') \prod_{k=1}^{K} p(\beta_k) d\beta_k \prod_{l=K+1}^{K'} \hat{p}(\beta'_l) d\beta'_k,
\end{aligned} \tag{E.10}$$

and similarly, it follows from Equation (E.9) that

$$\begin{aligned}
&\int_{A_{K'}} \int_{A_K} \psi_{K'}(d\boldsymbol{u}'_{K'}) P\left((K', \boldsymbol{u}'_{K'}), (K, d\boldsymbol{u}_K)\right) r\left((K', \boldsymbol{u}'_{K'}), (K, \boldsymbol{u}_K)\right) \\
&= \int_{A_K \cap U'_K} \int_{B_{K'}} \frac{p(\mathcal{D}|x'_1, \ldots, x'_{K'})}{p(\mathcal{D})} \pi(K') \pi\left(dx'_1, \ldots, dx'_{K'}\right) \times j_{K'}(K) \\
&\qquad r\left((K', \boldsymbol{u}'_{K'}), (K, \boldsymbol{u}_K)\right) \prod_{k=1}^{K'} p(\beta'_k) d\beta'_k,
\end{aligned} \tag{E.11}$$

where additionally we have changed the order of integration to $dx'_1 d\beta'_1 \ldots dx'_{K'} d\beta'_{K'}$. A couple of observations are worth stressing at this point. First,

$$\pi\left(dx_1, \ldots, dx_K, dx'_{K+1}, \ldots, dx'_{K'}\right)$$

in Equation (E.10) and

$$\pi\left(dx'_1, \ldots, dx'_{K'}\right)$$

[2]We note that $B_{K'}$ always exists because $A_{K'}$ is a multidimensional interval.

in Equation (E.11) are the same joint probability measure (as per our stochastic process prior), and the integrands in both Equations are integrated over the same domain. Second, the same holds for $\prod_{k=1}^{K} p(\beta_k)d\beta_k$ and $\prod_{k=1}^{K} p(\beta'_k)d\beta'_k$. Consequently, for detailed-balance (Equation (E.7)) to hold, it is sufficient that the acceptance probability of the Hastings move be such that

$$r\left((K, \boldsymbol{u}_K), (K', \boldsymbol{u}'_{K'})\right) p(\mathcal{D}|x_1, \ldots, x_K)\pi(K)j_K(K') \prod_{l=K+1}^{K'} \hat{p}(\beta'_l) \tag{E.12}$$

be equal to

$$r\left((K', \boldsymbol{u}'_{K'}), (K, \boldsymbol{u}_K)\right) p(\mathcal{D}|x'_1, \ldots, x'_{K'})\pi(K')j_{K'}(K) \prod_{l=K+1}^{K'} p(\beta'_l). \tag{E.13}$$

It is easy to verify that the acceptance probabilities of Equations (E.3) and (E.4) indeed satisfy this requirement, which concludes the proof of detailed-balance.

■

Appendix F

Derivations of Chapter 9

F.1 Pseudo-RKHS Completion of Neural Networks

In this section we prove Theorem 9.2, which we recall below.

Theorem F.1 *(Pseudo-RKHS completion of neural networks) Let $\mathcal{X}$ be a compact subset of a measurable space. Let*

$$\mathcal{F} = \left\{ f_\theta : \mathcal{X} \to \mathbb{R}, \quad \theta \in \mathbb{R}^N, \ N \in \mathbb{N}^* \right\}$$

be the hypothesis space of functions expressed by a neural network with weight parameters $\theta \in \mathbb{R}^N$, and Lipschitz activation function(s). There exists a pseudo reproducing kernel Hilbert space $\hat{\mathcal{H}}_k$, with continuous and bounded reproducing kernel k, such that

$$\mathcal{F} \subseteq \hat{\mathcal{H}}_k.$$

Proof The proof goes as follows. We consider a neural network with Lipschitz activation function(s) on a compact set, whose hypothesis space of candidate functions we denote $\mathcal{F}$. Firstly, we will construct a stochastic process h, whose space of paths, say $\mathcal{H}$, contains $\mathcal{F}$. Secondly, we will prove that $\mathcal{H}$ is included in the pseudo-RKHS whose reproducing kernel k is the covariance function of h. Finally, we will prove that k is continuous and bounded.

Construction of h: To obtain h we extend the neural network with an additional scaling (output) layer, mapping the output of the neural network, say $f_\theta(x)$, to $\alpha f_\theta(x)$ for some constant α, and we place i.i.d. standard Gaussian priors on all $(N+1)$ parameters. It is easy to note the space of functions encoded by the extended neural network

contains $\mathcal{F}$, the space of functions encoded by the original neural network, which is recovered as the special case $\alpha = 1$. Moreover, because the support of the (Gaussian) prior over the $(N+1)$ parameters is $\mathbb{R}^{N+1}$, the space of paths of h, namely $\mathcal{H}$, is the space of functions expressed by the extended neural network. Consequently, $\mathcal{F} \subset \mathcal{H}$.

$\mathcal{H}$ is contained in the pseudo-RKHS with reproducing kernel k: To prove that $\mathcal{H}$ is contained in a pseudo-RKHS, we will use the Karhunen-Loève expansion theorem (Loève, 1963, §37.5-B), which we recall below.

Theorem F.2 *(Karhunen-Loève) Let $(f(x))_{x\in\mathcal{X}}$ be a zero-mean square integrable stochastic process defined over a probability space $(\Omega, F, \mathbb{P})$, and indexed over a compact set $\mathcal{X} \subset \mathbb{R}^d$, with continuous covariance function $k_f(x, y)$. Then k_f is a Mercer kernel, and letting e_i be an orthonormal basis of $L^2(\mathcal{X})$ formed by the eigenfunctions of the integral operator*

$$T_{k_f}(g) = \int_{\mathcal{X}} k_f(x, .)g(x)dx$$

with respective eigenvalues λ_i, f admits the following representation:

$$f(x) = \sum_{i=1}^{\infty} Z_i e_i(x) \tag{F.1}$$

where the convergence is in the mean-square sense, uniform in x, and

$$Z_i = \int_{\mathcal{X}} f(x)e_i(x)\, dx.$$

Furthermore, the random variables Z_i have zero-mean, are uncorrelated and have variance λ_i

$$E\left(Z_i\right) = 0,\ \forall i \in \mathbb{N} \qquad \textit{and} \qquad E\left(Z_i Z_j\right) = \delta_{ij}\lambda_j,\ \forall i, j \in \mathbb{N}.$$

Corollary F.3 *If the stochastic process f in the Karhunen-Loève theorem is of the form $f(.; \theta(\omega))$, where the randomness occurs solely through a $\mathbb{R}^{N+1}$-valued Gaussian random variable θ, and if the function $\theta \to f(x, \theta)$ is Lipschitz for every $x \in \mathcal{X}$, then for every random event $\omega \in \Omega$, the resulting path $f(.; \theta(\omega))$ lies in the pointwise completion of the span of $\{\dots, e_i, \dots\}$*

$$\forall \omega \in \Omega,\ f(.; \theta(\omega)) \in \overline{span\left(\{\dots, e_i, \dots\}\right)},$$

which is also the pseudo-RKHS with reproducing kernel k.

Proof When the sum $\sum_{i=1}^{+\infty} Z_i(\theta)e_i(x)$ can be rewritten as a sum with a finite number of non-zero terms, it is easy to see that the convergence in Equation (F.1) is also in the pointwise almost-everywhere sense. Moreover, in the alternative case, it is well known that mean-square convergence of a sequence of random variables, for instance $\sup_{x\in\mathcal{X}} |f(x;\theta) - \sum_{i=1}^{n} Z_i(\theta)e_i(x)| \underset{n\to\infty}{\to} 0$, implies almost sure convergence of a subsequence to the same limit, here $\sup_{x\in\mathcal{X}} |f(x;\theta) - \sum_{i=1}^{s(n)} Z_i(\theta)e_i(x)| \underset{n\to\infty}{\to} 0$, where $s(n)$ is an increasing unbounded sequence. Hence, f is almost surely the uniform (and consequently also pointwise) limit of functions in the pseudo-RKHS, and therefore it almost surely lies in the pseudo-RKHS itself. Finally, we note that, by absolute continuity of a multivariate Gaussian probability measure with respect to the corresponding Lebesgue's measure, for any $x \in \mathcal{X}$, $\sum_{i=1}^{s(n)} Z_i(\theta)e_i(.)$ can only fail to converge to $f(.;\theta)$ for a set of θ, namely $\bar{\Theta} \subset \mathbb{R}^{N+1}$, that is of null Lebesgue's measure. Of course if $\bar{\Theta}$ is empty, then the proof of the corollary is over. Otherwise, let $\bar{\theta} \in \bar{\Theta}$ and $\mathcal{B}(\bar{\theta}, \epsilon)$ be the ball centred at $\bar{\theta}$ with radius $\epsilon > 0$. Recalling that $\mathcal{B}(\bar{\theta}, \epsilon)$ has non-null Lebesgue's measure, it follows that $\mathcal{B}(\bar{\theta}, \epsilon) \not\subseteq \bar{\Theta}$, and consequently we may find $\theta_\epsilon \in \mathcal{B}(\bar{\theta}, \epsilon)$ such that

$$\forall x \in \mathcal{X},\ f(x;\theta_\epsilon) = \lim_{n\to\infty} \left(\sum_{i=1}^{s(n)} Z_i(\theta_\epsilon)e_i(x) \right).$$

Letting $\epsilon \to 0$ and by continuity of $\theta \to f(x;\theta)$, it follows that

$$\forall x \in \mathcal{X},\ f(x;\bar{\theta}) = \lim_{\epsilon\to 0} f(x;\theta_\epsilon),$$

which, as the pseudo-RKHS is pointwise complete, implies that $f(.;\bar{\theta})$ is also in the pseudo-RKHS. This concludes the proof that

$$\forall \omega \in \Omega,\ \ f(.;\theta(\omega)) \in \overline{\text{span}\left(\{\ldots, e_i, \ldots\}\right)}.$$

■

In order to use Corollary (F.3) to prove that $\mathcal{H}$ is contained in the pseudo-RKHS with reproducing kernel the covariance function of h, namely k, we need to verify that h is indeed mean-zero and that k is continuous. The mean of h is obtained by a direct application of the law of total expectation:

$$\forall x \in \mathcal{X},\ \ E(h(x)) = E\left(\alpha f_\theta(x)\right) = E\left(E\left(\alpha f_\theta(x) | f_\theta(x)\right)\right) = E\left(E\left(\alpha\right) f_\theta(x)\right) = 0,$$

where the last equality stems from the fact that α is mean-zero. To prove that k is continuous, we prove h is mean-square continuous[1]. To prove that h is mean-square continuous we use the following lemma.

Lemma F.4 *Let $(h(x))_{x\in\mathcal{X}}$ be a mean-square continuous $\mathbb{R}^p$-valued stochastic process and $f:\mathbb{R}^p\to\mathbb{R}$ a Lipschitz function. Then $(f(h(x)))_{x\in\mathcal{X}}$ is a mean-square continuous stochastic process.*

Proof f being Lipschitz means that

$$\exists C>0, \text{ s.t. } \forall x, dx\in\mathbb{R}^p,\ |f(h(x+dx))-f(h(x))|^2\leq C||h(x+dx)-h(x)||^2.$$

The result follows from taking the expectation on both sides, letting dx go to 0, and using the mean-square continuity of h. ■

We apply this lemma to each neuron in h, recursively from the first layer to the output layer, to prove that the output of each neuron in h, including the output neuron (i.e. h itself), is a mean-square continuous stochastic process.

k is bounded: By the Cauchy-Schwartz inequality,

$$|k(x,y)|\leq\sqrt{Var(h(x))Var(h(y))},$$

so that to prove the boundedness of k, we may prove the boundedness of $Var(h(x))$. Moreover, by applying the law of total variance on $h(x)$, conditioning on $f(x)$, we get

$$Var(h(x))=E(Var(h(x)|f(x)))+Var(E(h(x)|f(x)))=E\left(f^2(x)\right).$$

As previously discussed, the output of every neuron in our random neural network, including $f(x)$, which is the output of the next to last neuron, is a mean-square continuous stochastic process. Consequently,

$$x\to E\left(f^2(x)\right):=Var\left(f(x)\right)+E\left(f(x)\right)^2$$

is contunious. As $\mathcal{X}$ is compact, $x\to E\left(f^2(x)\right)$ is bounded, which concludes the proof.

Remark F.5 *The idea of placing i.i.d. standard normal priors on weigths in a neural network was used in Cho and Saul (2009) to construct the arc-cosine kernel. However,*

[1]We recall that a second order stochastic process is mean-square continuous if and only if its mean and covariance functions are continuous (see for instance Hsing and Eubank, 2015, Th. 7.3.2).

the author did not study the link between the RKHS induced by the arc-cosine kernel, and the neural network it aims at mimicking, and restricted themselves to a single-layer network with Heaviside activation function, in the limit where the number of nodes is infinite.

■