Covariance Kernels for Fast Automatic Pattern Discovery and Extrapolation with Gaussian Processes

Andrew Gordon Wilson

Trinity College

University of Cambridge

This dissertation is submitted for the degree of

Doctor of Philosophy

January 2014

Acknowledgements

I thank Zoubin Ghahramani for always believing in me, and for granting me the freedom to pursue my own research direction. I am happy to have had such a good hearted and inspiring supervisor. I am also thankful to my advisor Carl Edward Rasmussen, whose dedication to science is inspiring.

The machine learning group at Cambridge has been a fantastic learning environment. I have learned so much from talking with others. I specifically thank Rich Turner, David Knowles, Sinead Williamson, John Cunningham, Ryan Turner, Matt Shannon, and Yunus Saatchi.

I have enjoyed my time at Trinity College. The stimulating conversations, wonderful music, beautiful architecture and culture are unique and unforgettable. Alex Gabrovsky, Charles Drummond, Adam Sproat, Robin Stephenson, Lennart Stern, and Dale Schwartz have particularly made my time there memorable.

My earliest memories are of my father excitedly taking me to observatories, showing me experiments, and amazing me with modern physics and the mysteries of the universe. I have since dedicated my life to science, and to understanding those mysteries. I am forever grateful to Cara, Jeremy, John, and my family, Gordon, Patricia, Josh, Willow, Lime, Leaf, Rain, Daisy, and Laska, for profoundly enriching my life, and for their constant good nature, unwavering support and interest.

I dedicate this thesis especially to Cara, the love of my life.

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration except where specifically indicated in the text. This disseration does not exceed seventy-one thousand five hundred words in length. No parts of this dissertation have been submitted for any other qualification.

This dissertation contains 69,987 words and 39 figures.

Covariance Kernels for Fast Automatic Pattern Discovery and Extrapolation with Gaussian Processes

Andrew Gordon Wilson

Abstract

Truly intelligent systems are capable of pattern discovery and extrapolation without human intervention. Bayesian nonparametric models, which can uniquely represent expressive prior information and detailed inductive biases, provide a distinct opportunity to develop intelligent systems, with applications in essentially any learning and prediction task.

Gaussian processes are rich distributions over functions, which provide a Bayesian nonparametric approach to smoothing and interpolation. A covariance kernel determines the support and inductive biases of a Gaussian process. In this thesis, we introduce new covariance kernels to enable fast automatic pattern discovery and extrapolation with Gaussian processes.

In the introductory chapter, we discuss the high level principles behind all of the models in this thesis: 1) we can typically improve the predictive performance of a model by accounting for additional structure in data; 2) to automatically discover rich structure in data, a model must have large support and the appropriate inductive biases; 3) we most need expressive models for large datasets, which typically provide

more information for learning structure, and 4) we can often exploit the existing inductive biases (assumptions) or structure of a model for scalable inference, without the need for simplifying assumptions.

In the context of this introduction, we then discuss, in chapter 2, Gaussian processes as kernel machines, and my views on the future of Gaussian process research.

In chapter 3 we introduce the Gaussian process regression network (GPRN) framework, a multi-output Gaussian process method which scales to many output variables, and accounts for input-dependent correlations between the outputs. Underlying the GPRN is a highly expressive kernel, formed using an adaptive mixture of latent basis functions in a neural network like architecture. The GPRN is capable of discovering expressive structure in data. We use the GPRN to model the time-varying expression levels of 1000 genes, the spatially varying concentrations of several distinct heavy metals, and multivariate volatility (input dependent noise covariances) between returns on equity indices and currency exchanges which is particularly valuable for portfolio allocation. We generalise the GPRN to an adaptive network framework, which does not depend on Gaussian processes or Bayesian nonparametrics; and we outline applications for the adaptive network in nuclear magnetic resonance (NMR) spectroscopy, ensemble learning, and change-point modelling.

In chapter 4 we introduce simple closed form kernels for automatic pattern discovery and extrapolation. These spectral mixture (SM) kernels are derived by modelling the spectral density of kernel (its Fourier transform) using a scale-location Gaussian mixture. SM kernels form a basis for all stationary covariances, and can be used as a drop-in replacement for standard kernels, as they retain simple and exact learning and inference procedures. We use the SM kernel to discover patterns and perform long range extrapolation on atmospheric CO_2 trends and airline passenger data, as well as on synthetic examples. We also show

that the SM kernel can be used to automatically reconstruct several standard covariances. The SM kernel and the GPRN are highly complementary; we show that using the SM kernel with the adaptive basis functions in a GPRN induces an expressive prior over non-stationary kernels.

In chapter 5 we introduce GPatt, a method for fast multidimensional pattern extrapolation, particularly suited to image and movie data. Without human intervention – no hand crafting of kernel features, and no sophisticated initialisation procedures – we show that GPatt can solve large scale pattern extrapolation, inpainting, and kernel discovery problems, including a problem with 383,400 training points. GPatt exploits the structure of a spectral mixture product (SMP) kernel, for fast yet exact inference procedures. We find that GPatt significantly outperforms popular alternative scalable Gaussian process methods in speed and accuracy. Moreover, we discover profound differences between each of these methods, suggesting expressive kernels, nonparametric representations, and scalable exact inference are useful in combination for modelling large scale multidimensional patterns.

The models in this dissertation have proven to be scalable and with greatly enhanced predictive performance over the alternatives: the extra structure being modelled is an important part of a wide variety of real data – including problems in econometrics, gene expression, geostatistics, nuclear magnetic resonance spectroscopy, ensemble learning, multi-output regression, change point modelling, time series, multivariate volatility, image inpainting, texture extrapolation, video extrapolation, acoustic modelling, and kernel discovery.

Contents

List of Figures

Chapter 1

Introduction

Machine learning is fundamentally about developing intelligent systems, which in turn is about developing statistical models for automatic pattern discovery and extrapolation. In order for a statistical model to learn and make decisions without human intervention, the model must be able to discover patterns, and extrapolate those patterns to new situations. It is therefore not surprising that early machine learning models, such as the perceptron (Rosenblatt, 1962), were inspired by a model of a neuron (McCulloch and Pitts, 1943); a model must have good inductive biases (assumptions) to extract useful information from data, and human inductive biases – encoded in neural architectures – have been finely tuned over millions of years of evolution. Later, Rumelhart et al. (1986) inspired hope that it would be possible to develop intelligent agents with models like neural networks.

In short, the ability for a model to learn from data is determined by:

1. The support of the model: what solutions we think are a priori possible.
2. The inductive biases of the model: what solutions we think are a priori likely.

For example, if we are performing character recognition, the support of the model represents which characters we think are a priori possible, and the inductive biases are which characters we think are a priori likely.

The Bayesian framework represents uncertainty and expresses prior information using probability distributions. Therefore Bayesian models are naturally suited to learning representations in data and generalising these representations to new situations[1]. Indeed it has been suggested that the human ability for inductive reasoning – concept generalization with remarkably few examples – could derive from combining rich prior information with Bayesian inference (Steyvers et al., 2006; Tenenbaum et al., 2011; Yuille and Kersten, 2006).

Bayesian nonparametric models are Bayesian models where the number of meaningful parameters grows with the data; in other words, the expressive power of a Bayesian nonparametric model scales with the amount of available information. Until now, Bayesian nonparametic models, and Gaussian processes in particular, have largely been used for smoothing and interpolation on small datasets. However, these models can accommodate highly expressive priors, with large support, and detailed inductive biases, and thus have great potential for developing intelligent systems. Moreover, such expressive models are most valuable on large datasets, since more data typically provides more information to learn expressive structure.

Gaussian processes are rich distributions over functions, which provide a Bayesian nonparametric approach to smoothing and interpolation. The support and inductive biases of a Gaussian process are specified by an interpretable covariance kernel. In this thesis, we introduce new covariance kernels to enable fast automatic pattern discovery and extrapolation with Gaussian processes. These kernels can be placed in the context of a historical progression of statistical models in machine learning.

At the present time, we have quickly entered an era of *big data*: "The future of the human enterprise may well depend on Big Data", exclaimed West (2013), writing for Scientific American. Indeed recent machine learning efforts have focused on developing scalable models for large datasets, with notable results from deep neural architectures (Krizhevsky et al., 2012).

Neural networks first became popular in the 1980s because they allowed for adaptive basis functions, as opposed to the fixed basis functions in well known linear

[1]In this thesis, we refer to *features*, *representations*, and *patterns* interchangeably.

models. With adaptive basis functions, neural networks could automatically discover interesting structure in data, while retaining scalable learning procedures (Rumelhart et al., 1986). But this newfound expressive power came at the cost of interpretability and the lack of a principled framework for deciding upon network architecture, activation functions, learning rates, etc., all of which greatly affect performance (Rasmussen and Williams, 2006).

Following neural networks came the kernel era of the 1990s, where infinitely many fixed basis functions were used with finite computational resources via the *kernel trick* – implicitly representing inner products of basis functions using a kernel. Kernel methods are flexible, and often more interpretable and manageable than neural network models. For example, Gaussian processes can be used as rich prior distributions over functions with properties – smoothness, periodicity, etc. – controlled by an interpretable covariance kernel. Indeed Gaussian processes have had success on challenging non-linear regression and classification problems (Rasmussen, 1996).

Within the machine learning community, Gaussian process research developed out of neural networks research. Neal (1996) argued that since we can typically improve the performance of a model by accounting for additional structure in data, we ought to pursue the limits of large models. Using character recognition as an example, Neal (1996) writes "it will always be possible to improve performance at least a bit by taking account of further rare writing styles, by modeling the shapes of the less common forms of ink blots, or by employing a deeper analysis of English prose style in order to make better guesses for smudged letters". The first examples – taking account of further rare writing styles and irregular ink blots – suggest we ought to make the support of a model as large as is reasonably possible. The next example, regarding a deeper analysis of English prose, suggests that we ought to refine our inductive biases – e.g., the *distribution* of prior support – as much as possible, so that we can improve, for instance, our prior beliefs about how likely certain characters might be. And large models – infinite models, ideally – can be used to provide large support and expressive inductive biases.

Accordingly, Neal (1996) showed that Bayesian neural networks become Bayesian nonparametric Gaussian processes with a *neural network kernel*, as the number of hidden units approach infinity. Thus Gaussian processes as nonparametric kernel machines are part of a natural progression, with the flexibility to fit any dataset, automatically calibrated complexity (Rasmussen and Ghahramani, 2001; Rasmussen and Williams, 2006), easy and interpretable model specification with covariance kernels, and a principled probabilistic framework for learning kernel hyperparameters.

However, kernel machines like Gaussian processes are typically unable to scale to large modern datasets. Methods to improve scalability usually involve simplifying assumptions, such as finite basis function expansions (Lázaro-Gredilla et al., 2010; Le et al., 2013; Rahimi and Recht, 2007; Williams and Seeger, 2001), or sparse approximations using pseudo (aka inducing) inputs (Hensman et al., 2013; Naish-Guzman and Holden, 2007; Quiñonero-Candela and Rasmussen, 2005; Seeger et al., 2003; Snelson and Ghahramani, 2006). The assumptions of these methods are often suitable for scaling popular kernel functions, but we will see that these assumptions are not as suitable for highly expressive kernels, particularly when a large number of training instances provide an opportunity to extract sophisticated structure from the data.

Indeed popular covariance kernels used with Gaussian processes are not often expressive enough to capture rich structure in data and perform extrapolation, prompting MacKay (1998) to ask whether we had "thrown out the baby with the bathwater". In general, choice of kernel profoundly affects the performance of a Gaussian process – as much as choice of architecture affects the performance of a neural network. Gaussian processes are sometimes used as flexible statistical tools, where a human manually discovers structure in data and then hard codes that structure into a kernel. Typically, however, Gaussian processes are used with the Gaussian (squared exponential) or Matérn kernels by default. In either case, Gaussian processes are used as smoothing interpolators, only able to discover limited covariance structure. Likewise, multiple kernel learning (Gönen and Alpaydın, 2011) typically involves hand crafting combinations of Gaussian kernels for specialized applications, such as modelling low dimensional structure in high dimensional data, and is not intended for automatic pattern discovery and extrapolation.

In short, Gaussian processes (GPs) as distributions over functions have a rich history in statistics, dating back to O'Hagan (1978), and described in detail in Rasmussen and Williams (2006), Stein (1999), MacKay (1998), and Cressie (1993); however, GPs have almost exclusively been used with simple kernels, such as the Gaussian kernel, limited to smoothing and interpolation.

In this thesis we propose new covariance kernels for fast automatic pattern discovery and extrapolation with Gaussian processes (GPs). Every model in this thesis is not simply for interpolation or smoothing, but is intended to discover patterns (aka features) in data, for the purpose of scientific discovery.

In chapter 2, we introduce Gaussian processes as kernel machines. Anything that may presently be unclear will hopefully be clarified in this chapter. The experienced Gaussian process researcher, however, may wish to treat this chapter as a reference, at least up until section 2.6, where I describe my views on the future of Gaussian process research. In short, I believe that more expressive kernels will become mainstream, and when this happens, the powerful framework provided by Gaussian processes for learning the properties of a kernel will increasingly distinguish GPs from other kernel machines. I also believe that the most successful way to scale these expresive kernels will be through exploiting existing model structure, rather than through the now popular inducing input or finite basis function assumptions.

In chapter 3, we introduce a new regression framework, Gaussian Process Regression Networks (GPRNs), which unifies many of the themes identified in the previous chapter 2 as important future areas of Gaussian process research: multi-output regression, heteroscedasticity, non-Gaussian predictive distributions, kernel learning and pattern discovery, and exploiting structure for fast scalable inference. Underlying all the properties of the GPRN is a highly expressive covariance kernel, which is formed by combining the nonparametric flexibility of Gaussian processes with certain structural properties of neural networks. Moreover, we exploit the structure of the GPRN so that it can scale linearly with the number of output variables (responses), as opposed to the more common cubic scaling in typical GP based multi-output models.

The GPRN is applied primarily as a multi-output regression model, with the distinctive ability to model *input dependent* signal and noise correlations between multiple outputs. In particular, we apply the GPRN to modelling the expression levels of 1000 genes (1000 output variables), which all depend on time (the input), and which are all thought to be regulated by one or a small number of transcription factors. The GPRN is different than a standard GP based multi-output models in that it assumes these correlations are time-varying. We also apply the GPRN to predicting the concentrations of different heavy metals at desired spatial locations in the Jura Swiss mountain region. The GPRN is able to learn how the correlations between these heavy metals vary as a function of geographical location, uncovering various geological features, which could correspond, for example, to rivers that affect how different heavy metals are distributed in the region. Finally, we apply the GPRN to multivariate volatility modelling in econometrics, where there are multiple output variables with time varying noise correlations (the data are heteroscedastic)[1].

In the Gaussian process regression network, multiple responses share latent basis functions, which is why the responses (e.g. multiple genes or multiple heavy metals) are correlated. The correlations change with inputs (predictors), because the connections from responses to basis functions are themselves functions of the predictors, rather than constants. In other words, connections in undirected graphical models, typically used to represent neural networks, become functions. This idea of an *adaptive network* has analogues in biological learning, does not depend on Bayesian nonparametrics, and has widespread implications in machine learning and statistics. For instance, using an adaptive network one can easily modify a given regression or classification method to account for input dependent correlations between multiple responses, often for greatly improved predictive performance. At the end of chapter 3, we discuss this generalisation of GPRNs to adaptive networks, and outline novel applications in nuclear magnetic resonance spectroscopy, ensemble learning, and change-point modelling.

[1]In appendix A1 we introduce time series models and the concept of volatility, or input dependent noise. This appendix is valuable background material for chapter 3.

The GPRN and adaptive network framework is highly complementary with the simple closed form kernels we introduce in the following chapter 4. At a basic level, these kernels in chapter 4 can be used to automatically discover features such as periodicity, smoothness, etc., without having to a priori hard code these features into a model. These kernels are derived by modelling a spectral density – the Fourier transform of a kernel – with a scale-location Gaussian mixture, and form a basis for all stationary kernels. We show that these spectral mixture (SM) kernels can be used as a drop-in replacement for standard kernels, with major benefits in expressiveness and performance, while retaining simple analytic inference and learning procedures. We use the SM kernel to perform automatic pattern discovery and extrapolation on airline passenger data, CO_2 readings, and synthetic examples. We also show that the SM kernel can reconstruct ground truth standard covariances automatically. We then develop expressive processes over kernels by using Gaussian processes with spectral mixture kernels as adaptive basis functions in the GPRN framework. These processes are highly expressive – with support for essentially any possible kernel – but naturally allow us to express our inductive biases, for example, by being able to concentrate prior support on stationary kernels, if desired.

In chapter 5, we extend and unify spectral mixture kernels with fast exact inference techniques, for large scale multidimensional pattern extrapolation. In particular, we introduce a spectral mixture product (SMP) kernel, which is intended for multidimensional inputs. We then exploit the Kronecker structure of this kernel for fast and exact inference in a method called GPatt. GPatt inference is most suited to inputs which have some grid structure[1] – images, video, spatial statistics, etc. – and thus is not typically as natural for very high dimensional input data. However, grid structure is not required, and many of the experiments we perform are on significantly non-grid data. Without human intervention – no hand crafting of kernel features, and no sophisticated initialisation procedures – We show that GPatt can solve practically important large scale pattern extrapolation, inpainting, video extrapolation, and kernel discovery problems, including a problem with

[1]By *grid*, we mean the inputs $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_P \subset \mathbb{R}^P$. Note that this is not a regularly spaced grid.

$383,400$ training points. We find that GPatt significantly outperforms popular alternative scalable Gaussian process methods in speed and accuracy. Moreover, we discover profound differences between each of these methods, suggesting expressive kernels, fully nonparametric representations, and exact inference are useful in combination for modelling large scale multidimensional patterns. GPatt provides a fresh and highly general approach to multidimensional pattern discovery, enabling largely new applications, such as video extrapolation, which is considered an open problem (Guillemot and Le Meur, 2014).

Finally, in chapter 6, we summarize the high level conclusions and discuss future directions for the work in this thesis. I believe we will soon enter a new era of machine learning, where models are highly expressive, but also interpretable and manageable, and are scalable through simple inference and learning procedures which exploit existing model structure. I hope to contribute to this direction with the models in this thesis.

We conclude the present introductory chapter with a basic introduction to some ideas in regression – touching on parametric modelling, model complexity, and model selection, as well as Bayesian nonparametric modelling. This introduction is intended to establish fundamental modelling principles which guided in the development of all of the models in this thesis.

1.1 Parametric Modelling

Suppose we are interested in performing regression. In other words, we have access to a *training set* of observations (or more generally, outputs) $\boldsymbol{y} = (y(x_1), \dots, y(x_n))^\top$, evaluated at a set of known inputs $X = (x_1, \dots, x_n)^\top$, and we wish to predict $y(x_*)$ at a test input x_*. The outputs, for example, could be noisy CO_2 readings, taken at a set of input times X. We could, for instance, have access to yearly CO_2 readings from 1930 to 2013 and we may wish to predict yearly CO_2 concentrations from 2014 to 2020. The inputs x could also be multidimensional, e.g., $x \in \mathbb{R}^M$.

A conceptually simple approach to the regression problem is to guess the form of a function that could have generated the data, and then learn the parameters of this function so as to minimize the error of predictions made on the training set. If the training set appears simple, for example if it has a linear trend, we could use a simple linear function $f(x, \boldsymbol{w}) = \boldsymbol{w}^\top x$, parametrized by a vector of 'weights' $\boldsymbol{w}$. Or if the data set seems more complex, we could use a linear model with a vector of non-linear basis functions (sometimes called *features*) $\boldsymbol{\phi}(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_k(x))^\top$, where $f(x, \boldsymbol{w}) = \boldsymbol{w}^\top \boldsymbol{\phi}(x)$. It is convention to call this a linear regression model, because it is linear in parameter space. This model is **not** necessarily linear in the input space x. Any polynomial, for example, would be a linear regression model with basis functions $1, x, x^2, \ldots$. If we want yet more flexibility, we could use a neural network function with many hidden units, which is a non-linear function in both parameter and input space.

It is common to take as error the squared difference between what the model predicts the output should be, and what the output actually is:

$$E(\boldsymbol{w}) = \sum_{i=1}^{N} (f(x_i, \boldsymbol{w}) - y_i)^2 \,. \tag{1.1}$$

In Eq. (1.1), N denotes the number of data points in the training set. Taking the gradient of E with respect to $\boldsymbol{w}$, we can sometimes find a minimum either analytically, or using a method like gradient descent. When choosing an error measure, we have to carefully consider what it means for the model to perform well on the test set.

Another approach is to assume that the training output is noisy, so that $y(x) = f(x, \boldsymbol{w}) + \epsilon(x)$, where ϵ is a random variable representing noise, and then set the parameters $\boldsymbol{w}$ by maximizing the *likelihood* $p(\boldsymbol{y}|X, \boldsymbol{w})$. Commonly one takes $\epsilon \sim \mathcal{N}(0, \sigma^2)$, in which case

$$p(y|x, \boldsymbol{w}) = \mathcal{N}(y(x); f(x, \boldsymbol{w}), \sigma^2) \,, \tag{1.2}$$

$$p(\boldsymbol{y}|X, \boldsymbol{w}) = \prod_{i=1}^{N} \mathcal{N}(y(x_i); f(x_i, \boldsymbol{w}), \sigma^2) \,. \tag{1.3}$$

The setting of the parameters $\boldsymbol{w}$ that maximizes the likelihood in Eq. (1.3) is the same setting of parameters that minimizes the squared error function Eq. (1.1). However, this would not necessarily be the case with a different noise model. The noise model can therefore be used to interpret a chosen error measure. Moreover, in this second *maximum likelihood* framework we can also maximize with respect to σ^2 to get a sense of the noise level.

Both of these techniques are prone to *overfitting*, which means that $f(x, \boldsymbol{w})$ has low error on reconstructing the training set, but high error on the test data, often because it is modelling random artifacts in the training data, rather than just the underlying signal. The more expressive the regression function f, the more capable it is of fitting sophisticated trends, but also the more likely it is to overfit. One way to manage overfitting is to introduce a complexity penalty term in the expression for the likelihood. For example, we may write a penalized log likelihood as

$$\log p(\boldsymbol{y}|X, \boldsymbol{w}) \propto \overbrace{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(f(x_i, \boldsymbol{w}) - y_i^2)}^{\text{model fit}} \overbrace{-\lambda \boldsymbol{w}^\top \boldsymbol{w}}^{\text{complexity penalty}} \quad . \tag{1.4}$$

The first term is indeed a scaled version of the squared error metric in Equation (1.1). Had we used, for example, a Laplace distribution rather than a Gaussian distribution to model the noise, we would have recovered an absolute value error metric. The complexity penalty discourages weights from becoming too large. Introducing a complexity penalty into the likelihood is called *regularization*. It is difficult to know how we should penalize complexity (what term should we introduce?). It is also uncertain how much we should penalize complexity; for example, what should we do with the parameter λ, since setting $\lambda = 0$ will trivially maximize likelihood? Indeed complexity itself is not even well defined or intuitive (and is discussed more in sections 1.2 and 2.3.2). *Cross-validation* can be used to set λ, in a procedure where parameters are trained with various settings of λ, and then the setting of the parameters with the lowest error on a held out validation set is used to learn λ. It can be difficult, however, to know how to choose the validation set(s) and how many settings of λ to try, particularly in situations where λ might be a vector.

A different way to protect from overfitting is to set a prior distribution on the weights, $p(\boldsymbol{w})$, such as $p(\boldsymbol{w}) = \mathcal{N}(0, \Sigma_w)$, and then use Bayes' rule to infer a posterior distribution over the weights $p(\boldsymbol{w}|\boldsymbol{y}, X)$:

$$\text{posterior} = \frac{\text{likelihood} \times \text{prior}}{\text{marginal likelihood}}, \quad p(\boldsymbol{w}|\boldsymbol{y}, X) = \frac{p(\boldsymbol{y}|X, \boldsymbol{w})p(\boldsymbol{w})}{p(\boldsymbol{y}|X)}. \tag{1.5}$$

The marginal likelihood is a normalizing constant which does not have an explicit dependence on the weights.

We are usually just interested in making predictions about y evaluated at a particular input x, not in the values of $\boldsymbol{w}$. The predictive distribution is given by

$$p(y|x, \boldsymbol{y}, X) = \int p(y|x, \boldsymbol{w})p(\boldsymbol{w}|\boldsymbol{y}, X)d\boldsymbol{w}. \tag{1.6}$$

Eq. (1.6) is an example of Bayesian model averaging, as we are performing a weighted average of each prediction made for each setting of the parameters $\boldsymbol{w}$, with the probability density of those parameters given the training data. The regression function is a random function, because the weight parameters are random variables. This method prevents overfitting (we are not simply using one model, but weighting models by their posterior probabilities), and gives a whole predictive distribution, rather than just a deterministic point estimate. This extra information is useful; for instance, it immediately tells us how confident we should be in any prediction we make. Moreover, the model average in Eq. (1.6) intuitively seems like the right thing to do: we don't have complete confidence that any one model generated the data. Ideally we would like $p(y|x, \boldsymbol{w})$ to include as many models as possible for different settings of $\boldsymbol{w}$, so we can weight the predictions of *all* feasible models by their posterior probabilities. This principle corresponds to wanting to have as much support as possible in our prior. Incidentally, the prior can be viewed as a *combination* of the functional form of $p(y|x, \boldsymbol{w})$ and the explicit prior distribution on the weights $\boldsymbol{w}$. Conventionally, one might assume the only prior is on $\boldsymbol{w}$; however, the functional form of a model reflects our prior assumptions at least as much as our uncertainty over the parameters of that model (the distribution on $\boldsymbol{w}$). In this sense, classical statisticians also use priors, and not just Bayesians. In other words, the prior on $\boldsymbol{w}$ induces a prior over functions parametrized by $\boldsymbol{w}$, and we are interested in the whole functions, not just the parameters of these functions.

In general, it is difficult to guess the parametric form of a function that could have generated our data. Surely any function we guess will be incorrect! Real world data is rarely generated by simple functions, and no data is likely to have been generated by, for example, a complicated neural network function. For example, could anyone be confident that $f(x, \boldsymbol{w}) = w_1 x_1 \tanh(w_2 x_2^{9/2} + w_3 \log(x_3^2) + w_4) + w_5 x_1^4$ truly describes some real data, even if it performs well on a training and test set?[1] And how are we to interpret the parameters of such models? And how are we to know what priors to place on the parameters? Moreover, the integral of Eq. (1.6) is only analytically tractable for highly limited parametric forms of $f(x, \boldsymbol{w})$, typically forcing one to resort to approximations (e.g., MCMC, variational Bayes, Laplace, EP, etc., discussed in Bishop (2006) and developed in chapter 3).

We will be more correct in guessing vague properties of the underlying function. We may have good intuitions about whether or not the function is smooth, periodic, or increasing. And it turns out that we just need these intuitions. Rather than perform inference in weight space, where we have little intuition, we can perform inference directly in function space, where we directly consider distributions over rich classes of functions. This *function space* view of regression is naturally suited to Gaussian processes, which we will discuss in detail in chapter 2.

1.2 Model Complexity and Model Selection

Figure 1.1 is often used to define model complexity (MacKay, 1992b; Rasmussen and Ghahramani, 2001). The vertical axis represents the likelihood of a given dataset $\boldsymbol{y}$ (indexed by X), having integrated away any parameters $\boldsymbol{w}$ from a model $\mathcal{M}$. For example, suppose we define a model $\mathcal{M}_1$ as the functional form $f_1(x, \boldsymbol{w})$, and a prior on parameters $p(\boldsymbol{w})$. Then the *marginal likelihood* of $\mathcal{M}_1$ is

$$p(\boldsymbol{y}|\mathcal{M}_1, X) = \int p(\boldsymbol{y}|f_1(x, \boldsymbol{w}))p(\boldsymbol{w})d\boldsymbol{w} \,. \tag{1.7}$$

We can interpret the marginal likelihood in Eq. (1.7) as the probability that we would generate a given dataset $\boldsymbol{y}$ if we were to randomly sample the parameters

[1]Here I have abused notation slightly so that x_1, x_2, x_3 are three components of a single vector input x.

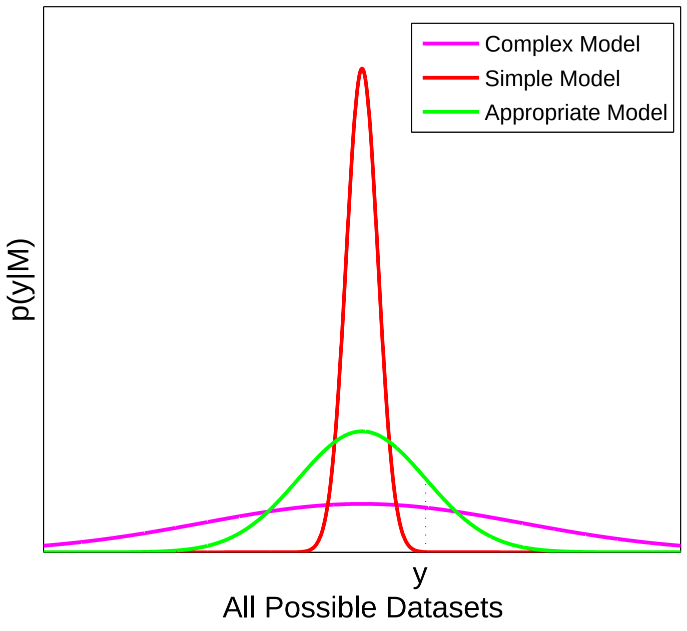


Figure 1.1: Model selection and complexity via marginal likelihood. We can interpret the marginal likelihood in Eq. (1.7) as the likelihood that we would generate a given dataset $\boldsymbol{y}$ if we were to randomly sample the parameters $\boldsymbol{w}$ and then condition on those parameters (e.g., $p(\boldsymbol{y}|y_1(x,\boldsymbol{w}))$). The marginal likelihood (evidence) is on the vertical axis, and possible data sets are on the horizontal. Models of three different complexities are shown. The simple model can only explain a small number of data sets, but it explains them well: they have a high likelihood. On the other hand, the complicated model can explain a large number of data sets, but somewhat poorly. A similar plot first appeared in MacKay (1992b).

$\boldsymbol{w}$ and then condition on those parameters (e.g., $p(\boldsymbol{y}|f_1(x,\boldsymbol{w}))$). The marginal likelihood is a proper probability distribution over $\boldsymbol{y}$; so models which give high marginal likelihood to a small number of datasets must give low probability to most other datasets, in order to normalize. Such models are labelled as simple models in Figure 1.1. Complex models are those models which spread their support over many datasets, but do not give very high probability to any given dataset.

While Figure 1.1 is an often used heuristic for defining complexity based on model evidence (marginal likelihood), and is useful for explaining the automatic complexity penalty in Gaussian process regression (section 2.3.2), it is not completely satisfactory. For example, if we can consider the marginal likelihood (evidence) for *any* model, then we can consider a model with point masses on the maximum likelihood solution of a large non-linear parametric model. Such a model will have high evidence on the dataset in question, and low evidence elsewhere, so would be considered simple. But such a model is at the greatest risk of overfitting the data. Alternatively, a parameter free model, which assigns equal probability to all possible datasets, would be extremely complex by Figure 1.1. Intuitively, complex models ought to be able to extract lots of information from the data, but the described parameter free model can't learn *anything*: predictions made by this model are independent of the data.

Furthermore, we would not want to perform model selection over the models of simple, intermediate, or high complexity in Figure 1.1, nor would we even want to perform a Bayesian model average over these models, because the number of discrete order models is small (countably infinite), compared to a continuous spectrum of models, and we do not believe any of these models generated the data.

I will exemplify this point by considering polynomials of different orders. Suppose we assume the observations $y(x)$ follow $p(y(x)|f(x)) = \mathcal{N}(y(x); f(x), \sigma^2)$, and consider polynomials of orders

$$f_0(x) = a_0\,, \tag{1.8}$$

$$f_1(x) = a_0 + a_1 x\,, \tag{1.9}$$

$$f_2(x) = a_0 + a_1 x + a_2 x^2\,, \tag{1.10}$$

$$\vdots \tag{1.11}$$

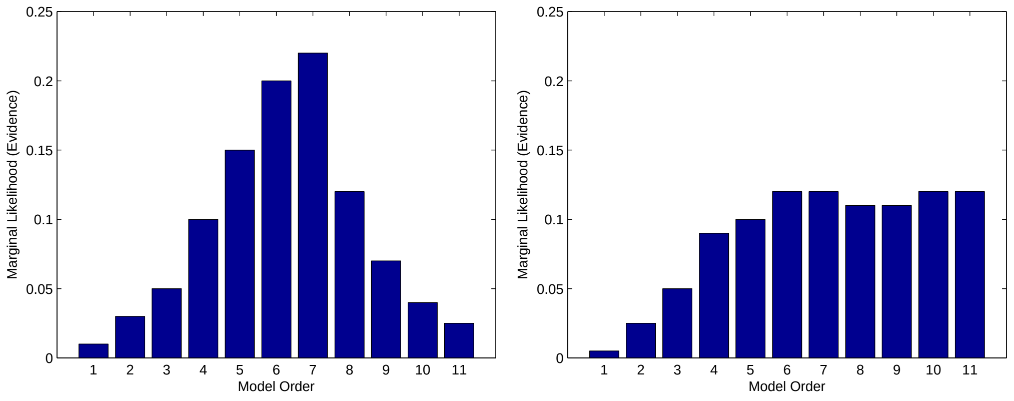
$$f_i(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_i x^i\,, \tag{1.12}$$

$$\vdots \tag{1.13}$$

$$f_J(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_J x^J\,. \tag{1.14}$$

Furthermore, suppose the actual function underlying the observed data is out of model class, as would always be the case in the real world. For instance, the data could be a step function, which cannot be exactly represented at all inputs $x \in \mathbb{R}$ by any finite order polynomial.

Introductory texts on machine learning and Bayesian inference such as Bishop (2006) often specify an isotropic prior over all parameters $p(\boldsymbol{a}) = \mathcal{N}(0, \sigma_f^2 I)$, and then analytically integrate away the parameters $\boldsymbol{a}$ in each of the model specifications above, to compute the evidence of the data $\boldsymbol{y}$ for each model order i, $p(\boldsymbol{y}|\mathcal{M}_i)$. The evidence is then plotted as a function of model order, as in Figure 1.2a to show an "Occam's hill" effect: typically the plot is unimodal, with a model of intermediate order giving the highest marginal likelihood. Like in Figure 1.1, this effect is used to show that the evidence favours the "simplest" model that describes the data – that model selection through marginal likelihood, in other words, embodies "Occam's razor".



(a) Isotropic Gaussian Prior (b) Anisotropic Gaussian Prior

Figure 1.2: Marginal likelihood as a function of model order. These plots are for data outside the model class – e.g., modelling a step function with finite order polynomials. These plots are based on the plots in Figures 2 and 4 in Rasmussen and Ghahramani (2001). a) With an isotropic Gaussian prior on the parameters, the marginal likelihood clearly favours a model of order 6. This effect is sometimes called "Occam's hill", and is also illustrated in Bishop (2006). b) When the variance of a Gaussian prior on parameters scales such that high order terms are given less prior probability mass – a scaling that is typically favoured by and can be learned from the data – then there is no Occam's hill; instead, there is roughly an asymptote towards infinite order models.

However, in actuality, we don't believe any finite order polynomial will describe real data (our data will always be out of model class), but we know that higher order polynomials can describe a larger variety of functions and over greater intervals. A polynomial of order i encompases all polynomials of order less than i. And a polynomial of arbitrary order can approximate any real valued continuous function on a closed interval to arbitrary accuracy (Weierstrass, 1885). In short, we know we can get *closer* to describing the true data generating process with a high order polynomial than a low order polynomial.

So we should not generally compute the evidence of multiple polynomials of different orders, and use these evidences to perform a Bayesian model average. We should instead use as big a polynomial as is computationally feasible. But Figure 1.1 suggests that such a polynomial would have low model evidence. How do we resolve this inconsistency? It turns out that, in this instance, "Occam's hill" is simply an artifact of a bad prior. If we let the variance on coefficient a_i scale with i, e.g. $p(a_i) = \mathcal{N}(0, i^{-\gamma} I)$, and learn the scaling γ from the data by maximizing the likelihood as a function of γ only (for a large, e.g. 200th, order model), then the evidence plot looks fundamentally different as a function of model order: there is now an asymptote towards infinite order models, as in Figure 1.2b.[1]

A similar result would be obtained from integrating away γ through sampling. The *data* are telling us to use an infinite order model, which harmonizes with our prior reasoning that infinite models will be closer to describing the true data generating process, which is undoubtedly outside of our model class. In a sense γ scales the complexity of the model – the higher gamma, the more low order models are favoured. One can plot samples from a high order polynomial, varying γ, to see how γ affects the induced distribution over functions: indeed, we ultimately care about the functions that model data and not the parameters in a parametric model, so it is sometimes more natural to directly consider functions at a high level, rather than the parameters – an idea which is naturally compatible with the Gaussian process framework presented in chapter 2.

The procedure that most honestly reflects our prior beliefs is therefore to use as high an order polynomial as possible, and then perform Bayesian inference in that model. As we will discuss in the next section, Bayesian nonparametric models

[1]Kass and Raftery (1995) and Kass and Greenhouse (1989) contain general discussions of the sensitivity of *Bayes factors* (the ratio of marginal likelihoods under two different models) to the choice of prior. For example, one must be careful when using improper priors when performing Bayesian model comparison.

allow us to work with infinite order models – containing support, for example, for polynomials of all order – with finite amounts of computation, and no over-fitting. These models allow us, therefore, to properly reflect our beliefs, and side-step conventional model selection.

So how should we define complexity? Recall that by Figure 1.1 a parameter free model would be considered complex, but could not learn anything from the data, which goes against our intuitive notions of complexity.

We therefore argue that we should tie the notion of complexity to the amount of information that can be learned from data (and we ought to use the most complex models possible). Murray and Ghahramani (2005) propose to use the mutual information between the training data and the predictions made by a model to define the "information capacity" of the model as

Definition 1.2.1. *The capacity (flexibility) of a model $\mathcal{M}_i$ can be defined as the mutual information between the data $\boldsymbol{y}$ (at N locations X) and predictions made by the model $\boldsymbol{y}_*$ (at test locations X_*)*

$$I_{i,N} = \sum_{\boldsymbol{y},\boldsymbol{y}_*} p(\boldsymbol{y}, \boldsymbol{y}_* | \mathcal{M}_i) \log \frac{p(\boldsymbol{y}, \boldsymbol{y}_* | \mathcal{M}_i)}{p(\boldsymbol{y}|\mathcal{M}_i) p(\boldsymbol{y}_*|\mathcal{M}_i)} \tag{1.15}$$

If we condition on the training data $\boldsymbol{y}$, and consider a continuous set of test points $\boldsymbol{y}_*$, then Eq. (1.15) becomes

$$I_{i,N} = p(\boldsymbol{y}) \int p(\boldsymbol{y}_*|\boldsymbol{y}) \log \frac{p(\boldsymbol{y}_*|\boldsymbol{y})}{p(\boldsymbol{y}_*)} d\boldsymbol{y}_* \,. \tag{1.16}$$

If we are to use Def. 1.2.1 as a definition for complexity, then more complex models are capable of extracting more information from the training data.

It would be worthwhile, in the future, to further explore exactly what it means for a model to be complex. A complete definition should explicitly consider rates of learning as well as a metric such as mutual information. Overall, one should use the models with the largest support imaginable. And what these models can learn from the data – and at what rate – is determined by their inductive biases.

1.3 Bayesian Nonparametric Models

I have argued that one ought to pursue the most expressive models possible, as the most expressive models typically reflect our honest beliefs about the real world. In other words, one can typically always improve predictions by accounting for additional structure in data. Moreover, in Bayesian modelling, one does not need to limit the expressiveness of a model for fear of over-fitting.

A parametric Bayesian model can be completely summarized with a finite set of parameters. Conversely, the number of parameters that determine the properties of a Bayesian nonparametric model depends on the amount of available data. That is, the information capacity of a Bayesian nonparametric model grows with the data, which is an intuitively desirable property: the more information we have seen, the more information the model ought to be able to express. In order to represent a whole function with a Bayesian nonparametric model, one would need an infinite number of parameters: for example, each function value $f(x_i)$ could be seen as a separate parameter for every $x_i \in \mathbb{R}^M$. Surprisingly, as shown in sections 2.3.1 and 2.4.3, it is possible to do inference and learning with Bayesian nonparametric methods using finite amounts of computational power. Such infinite models can exactly contain a wide range of conventional parametric models – for example, all of the polynomials given in section 1.2.

In short, we can get closer to reflecting our true beliefs using Bayesian nonparametric models, since 1) the information capacity of a Bayesian nonparametric model scales with the amount of available information, and 2) the large support of a Bayesian nonparametric model reflects the belief that we do not know the exact form of the function that generated any given dataset. Moreover, 3) Bayesian nonparametric methods can express the rich prior information that may be necessary for the remarkable generalisations we associate with human intelligence, and 4) when there is more available data, more information can be extracted by such expressive models. For these reasons, I believe Bayesian nonparametric models are naturally suited to pattern discovery, representation learning, and extrapolation on large datasets, and have much unexplored promise in this direction, given that these models (particularly Gaussian processes) are almost exclusively used for smoothing and interpolation on small datasets.

In section 2.6.1 I further discuss Bayesian nonparametrics in the context of future directions for Gaussian process research. Ghahramani (2013), Orbanz and Teh (2010) and Hjort et al. (2010) contain general expositions on Bayesian nonparametrics.

Chapter 2

Gaussian Processes

2.1 Introduction

In this thesis we wish to automatically discover rich structure in data with expressive covariance kernels.[1] Gaussian processes provide a uniquely interpretable and principled probabilistic framework for learning with kernels. A Gaussian process (GP) can be used as a distribution over functions, with support (prior probability) for essentially any function that could generate a given dataset. The distribution of this support – the properties of likely functions under a Gaussian process (smoothness, periodicity, etc.) – can be controlled by an interpretable covariance kernel. Indeed, "the problem of *learning* in Gaussian processes is exactly the problem of finding suitable properties for the covariance kernel" (Rasmussen and Williams, 2006). Gaussian processes as distributions over functions, with properties controlled by a kernel, have a rich history in statistics, starting with O'Hagan (1978) and described in detail in Rasmussen and Williams (2006), Stein (1999), MacKay (1998), and Cressie (1993).

Owing to their flexibility, interpretability, large support, consistency, simple exact learning and inference procedures, suitability for kernel learning, and impressive empirical performances (Rasmussen, 1996), Gaussian processes as kernel machines,

[1]We refer to *kernels*, *covariance kernels* and *covariance functions* interchangeably.

have steadily grown in popularity over the last decade.[1] However, despite their many appealing qualities, Gaussian processes as kernel machines are still not in widespread use. Their applications have mostly been limited to smoothing and interpolation, on datasets with fewer than 10000 points. Perhaps Gaussian processes have been held back by limited covariance kernels, computational intractability on large datasets (naively requiring $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ storage for a dataset of size N), and difficulties with multi-output regression and heteroscedasticity. We discuss each of these limitations – and some solutions – in sections 2.6.2, 2.6.3, 2.6.4, 2.6.5, as part of a larger discussion about the promising future of Gaussian processes (section 2.6), with pointers to new methodology presented in this thesis.

We begin with a self contained introduction to Gaussian process models. In section 1.1 we provided a general overview of parametric modelling, which serves to motivate 1) a function space, rather than weight space view of regression, and 2) non-parametric modelling. Both of these properties are naturally accommodated by a Gaussian process regression framework. I start with a general characterisation of GPs in section 2.2, which quickly moves from the weight space to function space views of modelling. We then discuss inference in section 2.3, covariance kernels in section 2.4, and a Bayesian nonparametric interpretation, my focus in this thesis, in section 2.6.1.

For concreteness, we will often be considering regression problems where we place a latent Gaussian process prior over functions $f(x)$ to model a set of observations, $\boldsymbol{y} = (y(x_1), \dots, y(x_n))^\top$, evaluated at a set of known inputs $X = (x_1, \dots, x_n)^\top$.[2,3] We wish to infer a posterior distribution over $f(x)$, and the predictive distribution of $y(x_*)$ at a test input x_*, given the observations $\boldsymbol{y}$. The observations, for example, could be measurements of a heavy metal, and the inputs could be spatial locations

[1]A *kernel machine* is any algorithm which makes use of a kernel, which is informally a measure of similarity between pairs of input points, often involving an inner product of basis functions in a 'feature space' (Bishop, 2006). The most well known kernel machine is perhaps the *support vector machine* (SVM) (Cortes and Vapnik, 1995). Section 2.4 has more information on kernels.

[2]We often use the terms *observations*, *targets*, *responses*, and *outputs* interchangeably.

[3]For notational convenience, we will often not explicitly write out conditional dependencies on the inputs X.

in $\mathbb{R}^2$. However, my considerations typically extend – without modification – to classification and unsupervised learning problems.

2.2 Gaussian Process Characterisation

Gaussian processes are ubiquitous. Even the basic linear model

$$f(x) = a_0 + a_1 x\,, \tag{2.1}$$

$$a_0 \sim \mathcal{N}(0,1), \quad a_1 \sim \mathcal{N}(0,1)\,, \tag{2.2}$$

is an example of a Gaussian process over f, an output variable, depending on an input variable $x \in \mathbb{R}^1$.[1]

A Gaussian process is a type of stochastic process, and informally, a stochastic process is a random function.[2] Placing a distribution over a_0 and a_1 induces a distribution over random linear functions $f(x)$. In Figure 2.1, we sample a_0 and a_1 from a standard Gaussian distribution and then condition on those samples to use Eq. (2.1) to sample from $f(x)$ for any x.

A Gaussian process is formally defined as follows:

Definition 2.2.1. *(Gaussian process). A Gaussian process is a collection of random variables, any finite number of which have a joint Gaussian distribution.*

In Equation (2.1), any function value $f(x)$ at any input x is marginally Gaussian, since Gaussian variables are closed under addition. Any pair of function values,

[1]In statistics, x is sometimes called a *predictor* variable or a *covariate*. In general we will consider $x \in \mathbb{R}^M$ for any natural number M.

[2]In the time series literature, a stochastic process is often defined as a random variable that evolves with time. In general, a stochastic process can depend on any arbitrary input, however. See Grimmett and Stirzaker (2001) for a more detailed but still accessible treatment of stochastic processes.

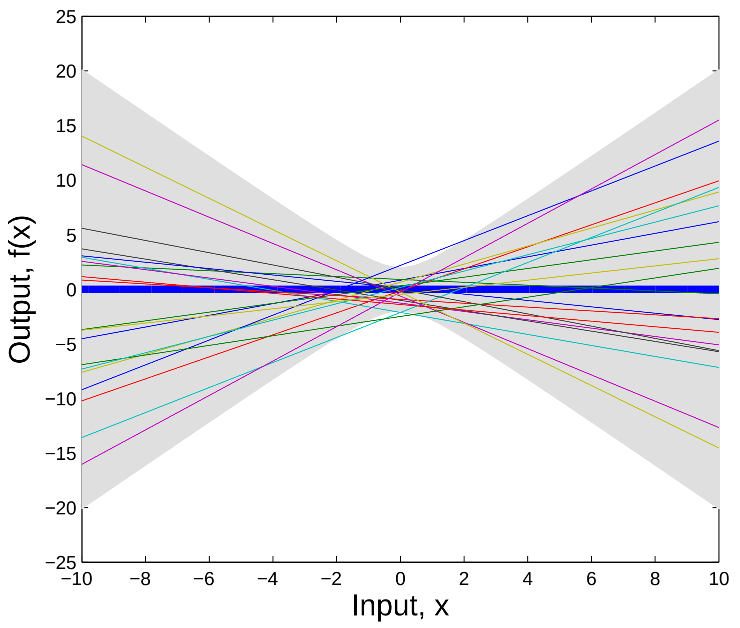


Figure 2.1: Sample functions drawn from a Gaussian process distribution over linear functions $f(x)$. The mean function $\mathbb{E}[f(x)]$ is shown in thick blue, and 95% of the mass of the distribution $(2\sqrt{\text{var}[f(x)]})$ is shown in gray shade. If we were to sample infinitely many functions from Eq. (2.1) and then average these functions, we would recover the mean function in thick blue. 95% of these sample functions would be contained within the gray shade. 20 sample functions are shown as thin lines in different colours. These functions are sampled at finitely many input points x and then the points are joined together to create a line in this figure (in this case, only two points would fully define a given sample, since these functions are straight lines).

$f(x_b)$ and $f(x_c)$, evaluated at a pair of input points, x_b and x_c, has covariance

$$\begin{aligned}\mathrm{cov}(f(x_b), f(x_c)) &= \mathbb{E}[f(x_b)f(x_c)] - \mathbb{E}[f(x_b)]\mathbb{E}[f(x_c)] && (2.3)\\ &= \mathbb{E}[a_0^2 + a_0a_1(x_b + x_c) + a_1^2 x_b x_c] - 0 && (2.4)\\ &= \mathbb{E}[a_0^2] + \mathbb{E}[a_1^2 x_b x_c] + \mathbb{E}[a_0a_1(x_b + x_c)] && (2.5)\\ &= 1 + x_b x_c + 0 && (2.6)\\ &= 1 + x_b x_c\,. && (2.7)\end{aligned}$$

Moreover, any pair $f(x_b), f(x_c)$ is jointly Gaussian, since each is a linear combination of the same a_0, a_1. An inductive argument can be used to show that any collection of function values is thus jointly Gaussian. Therefore any collection of function values in Eq. (2.1) has a joint Gaussian distribution with mean vector $\boldsymbol{\mu} = \mathbf{0}$ and $N \times N$ covariance matrix K,

$$[f(x_1), \dots, f(x_N)] \sim \mathcal{N}(\boldsymbol{\mu}, K)\,, \tag{2.8}$$

where $K_{ij} = 1 + x_i x_j$. Hence Eq. (2.1) defines a Gaussian process over $f(x)$.

In fact, *any* linear basis function model[1]

$$f(x) = \boldsymbol{w}^\top \boldsymbol{\phi}(x)\,, \tag{2.9}$$

with any Gaussian distribution over *weights* $\boldsymbol{w}$, and (potentially non-linear) basis functions $\boldsymbol{\phi}(x) = [\phi_1(x), \dots, \phi_J(x)]^\top$, is a Gaussian process,

$$f(x) \sim \mathcal{GP}(m(x), k(x, x'))\,, \tag{2.10}$$

entirely defined by its mean and covariance function

$$\begin{aligned} m(x) &= \mathbb{E}[f(x)]\,, && (2.11)\\ k(x, x') &= \mathrm{cov}[f(x), f(x')], && (2.12)\end{aligned}$$

for any pair of inputs x and x', in any arbitrary input space $x \in \mathcal{X}$. That is, any collection of function values, evaluated at any collection of input points, has a joint Gaussian distribution,

$$[f(x_1), \dots, f(x_N)] \sim \mathcal{N}(\boldsymbol{\mu}, K)\,, \tag{2.13}$$

[1]A linear basis function model in parameters $\boldsymbol{w}$ is any model which is linear in $\boldsymbol{w}$; the model does *not* also need to be linear in the inputs x.

with mean vector and covariance matrix

$$\boldsymbol{\mu}_i = \mathbb{E}[f(x_i)] \tag{2.14}$$

$$K_{ij} = k(x_i, x_j)\,. \tag{2.15}$$

The standard notation in Eq. (2.10) can be confusing: x' appears on the right hand side of the equation, but not on the left hand side. Eq. (2.10) simply means that the function evaluated at any finite set of input points can be expressed as Eq. (2.13). The kernel k operates on a pair of input points, x and x'. To avoid overloading the notation for the inputs x, it is often preferable to write the kernel as $k(x_a, x_b)$, where x_a and x_b are input points, or simply as k.

In the example of Eq. (2.2), $\boldsymbol{w} = (a_0, a_1)^\top$ and $\boldsymbol{\phi}(x) = (1, x)^\top$. In general, if $\boldsymbol{w}$ has distribution $\mathcal{N}(0, \Sigma_w)$, then

$$m(x) = \mathbb{E}[f(x)] = \boldsymbol{\phi}(x)^\top \mathbb{E}[\boldsymbol{w}] = 0\,, \tag{2.16}$$

$$k(x, x') = \mathbb{E}[f(x)f(x')] = \boldsymbol{\phi}(x)^\top \mathbb{E}[\boldsymbol{w}\boldsymbol{w}^\top]\boldsymbol{\phi}(x') = \boldsymbol{\phi}(x)^\top \Sigma_w \boldsymbol{\phi}(x')\,. \tag{2.17}$$

It is also straightforward to prove that if $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{\mu}, \Sigma_w)$, with arbitrary mean $\boldsymbol{\mu}$, then $\boldsymbol{w}^\top \boldsymbol{\phi}(x)$ is a Gaussian process.

A covariance function $k(x, x')$, also known as a *covariance kernel*, or simply a *kernel*, controls the properties of likely functions under a Gaussian process $f(x)$. In other words, the inductive biases of a Gaussian process model – whether we expect smooth functions, periodic functions, Brownian motion, etc. – are determined by a kernel.

We are ultimately more interested in – and have stronger intuitions about – the *functions* that model data than the weights $\boldsymbol{w}$ in a parametric model, and we can express those intuitions with a covariance kernel. Thus it is more natural to directly define a prior over *functions*, as in Eq. (2.10), and then perform inference in *function space*, rather than work in *weight space*, as in Eq. (2.9).

The popular Gaussian or *squared exponential* (SE) kernel,

$$k_{\text{SE}}(x, x') = a^2 \exp(-0.5||x - x'||^2/\ell^2)\,, \tag{2.18}$$

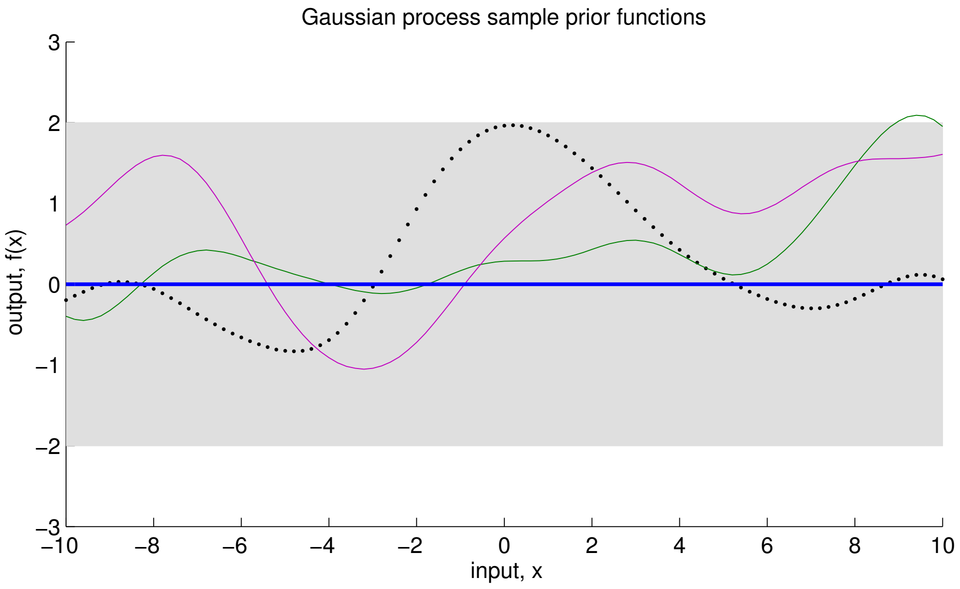


Figure 2.2: Sample functions from a Gaussian process prior with an SE kernel for $x \in \mathbb{R}^1$. A Gaussian process is sampled (black dots) at a given set of input locations. Two other sample functions are shown by purple and green curves; here the function values (dots) have been joined together. The Gaussian process mean function is shown in thick blue, and 95% of the mass of the Gaussian process about the mean (2 standard deviations) is shown in gray shade.

expresses the intuition that function values at nearby input points are more correlated than function values at far away input points.[1,2] Draws from a Gaussian process with an SE kernel are shown in Figure 2.2. The *length-scale* hyper-parameter ℓ determines how quickly Gaussian process functions vary in input space – how wiggly the functions in Figure 2.2 are. The *signal variance* parameter a^2 controls the variability of sample functions from the mean function – e.g. how concentrated the support of the Gaussian process prior around the mean function is. A Gaussian process with an SE kernel has support for *any* continuous function within an arbitrarily small band of width ϵ (Ghosal and Roy, 2006), and is an example of a truly nonparametric Gaussian process (section 2.6.1), derived from a linear regression model with an *infinite* number of basis functions (section 2.4.3).

[1] *Squared exponential* for Eq. (2.18) is a misnomer in common use: the distance $||x - x'||$ is squared, not the whole exponential. Neil Lawrence has proposed *exponentiated quadratic* as an alternative.

[2] We are now considering general input points $x \in \mathbb{R}^P$.

Indeed, by representing Gaussian process models in *function space*, using a mean function and covariance kernel, we can make predictions with models that have an infinite number of parameters (weights), in a finite amount of computational time! The "question of how we deal computationally with these infinite dimensional objects has the most pleasant resolution imaginable" (Rasmussen and Williams, 2006), which we discuss in the next section on inference with Gaussian processes.

2.3 Inference and Model Selection

2.3.1 Inference

Suppose we place a Gaussian process prior over functions $f(x) \sim \mathcal{GP}(m, k)$, to model a set of noisy targets $\boldsymbol{y} = (y(x_1), \dots, y(x_N))^\top$ at a set of inputs $X = (x_1, \dots, x_N)^\top$, where $x \in \mathbb{R}^P$.[1] We want to infer a posterior distribution over functions $f(x)$ and make predictions of $\boldsymbol{y}_*$ at a set of test inputs $X_* = (x_{*1}, \dots, x_{*M})^\top$. An *observation model* (sometimes equivalent to a noise model) relates the Gaussian process to the targets. For example, if the targets are the binary class labels $\{-1, +1\}$, we can choose the observation model $p(y(x) = 1|f(x)) = \sigma(f(x)) = 1/(1 + \exp[-f(x)])$. Alternatively, if the targets are real valued, we may assume that each target is equal to a latent Gaussian process plus Gaussian noise. For example,

$$y(x) = f(x) + \epsilon(x)\,, \tag{2.19}$$

$$\epsilon(x) \sim \mathcal{N}(0, \sigma^2)\,. \tag{2.20}$$

Equivalently, $p(y(x)|f(x)) = \mathcal{N}(y(x); f(x), \sigma^2)$. The predictive distribution $p(\boldsymbol{y}_*|\boldsymbol{y})$, omitting the input sets X and X_* for notational convenience, is given by

$$p(\boldsymbol{y}_*|\boldsymbol{y}) = \int p(\boldsymbol{y}_*|f(x))p(f(x)|\boldsymbol{y})df(x)\,, \tag{2.21}$$

$$p(f(x)|\boldsymbol{y}) \propto p(\boldsymbol{y}|f(x))p(f(x))\,. \tag{2.22}$$

[1]For illustrative purposes, we will often assume $x \in \mathbb{R}^1$ in figures.

The integral in Eq. (2.22) may seem daunting: how are we to access the Gaussian process $f(x)$ at every x in some potentially uncountably infinite input space $\mathcal{X}$, and even if we could, would the integral be analytically tractable?

2.3.1.1 Gaussian Likelihoods

With a Gaussian observation model, as in Eqs. 2.19 and 2.20, the integral in Eq. 2.22 is indeed analytically tractable! For notational simplicity, we assume a zero mean GP ($m(x) = 0$), let $\boldsymbol{f}_* = (f(x_{*1}), \ldots, f(x_{*M}))^\top$, and let $K(X, X_*)$ be an $N \times M$ matrix of covariances between the Gaussian process $f(x)$ evaluated at the training and test sets X and X_*, and likewise for $K(X, X)$, $K(X_*, X)$ and $K(X_*, X_*)$. We leave it as an exercise to show that the function $y(x)$ is a Gaussian process with covariance function $\mathrm{cov}(y_p, y_q) = k(x_p, x_q) + \sigma^2 \delta_{pq}$, where δ_{pq} is the Kronecker delta, such that $\mathrm{cov}(\boldsymbol{y}) = K(X, X) + \sigma^2 I$. The joint distribution over the targets (observations) $\boldsymbol{y}$, at training locations X, and the Gaussian process $\boldsymbol{f}_*$, evaluated at the test locations X_*, is thus given by

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{f}_* \end{bmatrix} \sim \mathcal{N}\left(\boldsymbol{0}, \begin{bmatrix} K(X, X) + \sigma^2 I & K(X, X_*) \\ K(X_*, X) & K(X_*, X_*) \end{bmatrix}\right) . \tag{2.23}$$

Since the Gaussian distribution is *consistent*, or has the *marginalization property*, examination of any set of Gaussian random variables does not change the distribution of any subset, and so we can trivially marginalize away any unneeded values of $f(x)$ to obtain the expression in Equation (2.23). Therefore – since we only ever need to query the Gaussian process at a finite number of points – we only need finite computational resources, even if the covariance function we are using is derived from an infinite basis function expansion, *and this holds even if we are not using a Gaussian observation model* (we discuss non-Gaussian likelihoods next in section 2.3.1.2). Given any hyperparameters[1] $\boldsymbol{\theta}$ of the covariance kernel, the joint distribution $p(\boldsymbol{f}_*, \boldsymbol{y})$, and marginal $p(\boldsymbol{y})$, we can apply Bayes' theorem to find the

[1]It is conventional to say that the data depend on parameters, in this case $f(x)$, and any parameters $f(x)$ depends on (in the next level of a hierarchical model) are called *hyperparameters*. E.g., Hyperparameters $\rightarrow$ Parameters $\rightarrow$ Data.

predictive distribution

$$\boldsymbol{f}_*|X_*, X, \boldsymbol{y}, \boldsymbol{\theta} \sim \mathcal{N}(\bar{\boldsymbol{f}}_*, \mathrm{cov}(\boldsymbol{f}_*)) \,, \tag{2.24}$$

$$\bar{\boldsymbol{f}}_* = K(X_*, X)[K(X,X) + \sigma^2 I]^{-1}\boldsymbol{y} \,, \tag{2.25}$$

$$\mathrm{cov}(\boldsymbol{f}_*) = K(X_*, X_*) - K(X_*, X)[K(X,X) + \sigma^2 I]^{-1} K(X, X_*) \,. \tag{2.26}$$

Letting $\boldsymbol{k}_*$ be a vector of covariances between the N training points and a test point x_*, $(\boldsymbol{k}_*)_i = k(x_*, x_i)$, the posterior distribution of a single output f_* at x_* is

$$f_*|x_*, X, \boldsymbol{y}, \boldsymbol{\theta} \sim \mathcal{N}(\bar{f}_*, \mathbb{V}[f_*]) \tag{2.27}$$

$$\bar{f}_* = \boldsymbol{k}_*^\top (K + \sigma^2 I)^{-1} \boldsymbol{y} \tag{2.28}$$

$$\mathbb{V}[f_*] = k(x_*, x_*) - \boldsymbol{k}_*^\top (K + \sigma^2 I)^{-1} \boldsymbol{k}_* \,. \tag{2.29}$$

The distribution over the targets, $p(\boldsymbol{y}_*|\boldsymbol{y})$, is Gaussian with the same moments as Equation (2.24), except with $\sigma^2 I$ added to the covariance in Equation (2.26). Moreover, we can use Equation (2.24) to query a noise free Gaussian process $f(x)$ at a set of points X_*, conditioned on the Gaussian process $\boldsymbol{f}$ at a set of points X, by setting $\sigma^2 = 0$ and $\boldsymbol{y} = \boldsymbol{f}$.

Figure 2.3 shows sample a) prior and b) posterior functions drawn from a Gaussian process with an SE kernel (Eq.(2.18)). We see that the functions look too quickly varying – too wiggily, too complex – for the data. We have set the length-scale hyperparameter ℓ too small. In order to automatically learn the rate of variation of these functions, we must perform model selection, discussed in section 2.3.2, to learn the hyperparameters $\boldsymbol{\theta}$ of a covariance kernel.

2.3.1.2 Non-Gaussian Likelihoods

If the likelihood $p(\boldsymbol{y}|\boldsymbol{f})$ is not Gaussian in $\boldsymbol{f}$ – for instance, if we are doing classification where we have an observation model $p(y(x) = 1|f(x)) = \sigma(f(x))$ – then the integral in Equation (2.22) is not analytic. To sample from the predictive distribution $p(y_*|\boldsymbol{y})$ it suffices to sample from $p(f_*|\boldsymbol{y})$ using a simple Monte carlo

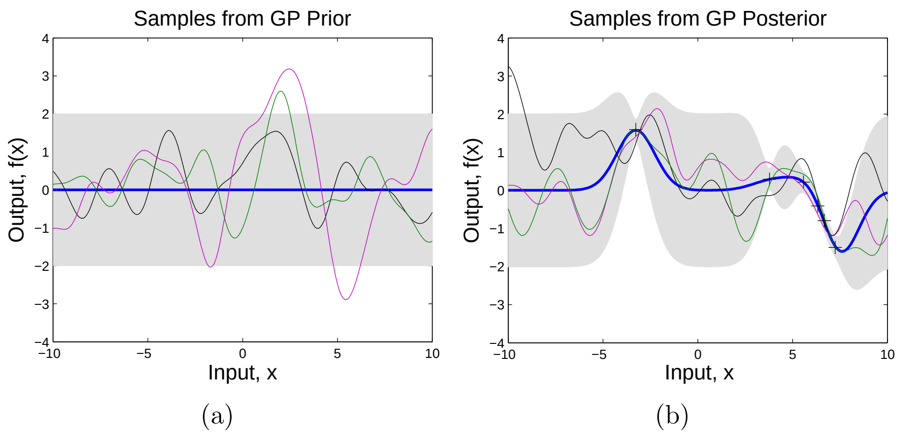


Figure 2.3: Sample a) prior and b) posterior functions drawn from a GP with an SE kernel with a too small length-scale. The length-scale ℓ is too small, causing sample functions to vary too quickly. The GP mean function is shown in blue, and 95% of the probability mass about the mean function is shown with gray shade. The data are denoted by black markers.

(SMC) sum[1],

$$p(y_*|\boldsymbol{y}) = \int p(y_*|f_*)p(f_*|\boldsymbol{y})df_*\,, \tag{2.30}$$

$$p(y_*|\boldsymbol{y}) = \lim_{J\to\infty}\frac{1}{J}\sum_{i=1}^{J} p(y_*|f_*^{(i)})\,, \qquad f_*^{(i)} \sim p(f_*|\boldsymbol{y})\,, \tag{2.31}$$

where Eq (2.31) is approximated using a finite value for J. However, typically there is an analytic relationship between $p(f_*|\boldsymbol{y})$ and $p(y_*|\boldsymbol{y})$. Either way, one must infer

$$p(f_*|\boldsymbol{y}) = \int p(f_*|\boldsymbol{f})p(\boldsymbol{f}|\boldsymbol{y})d\boldsymbol{f} \tag{2.32}$$

Equation (2.24) gives an exact Gaussian expression for $p(f_*|\boldsymbol{f})$ for $\sigma^2 = 0$ and $\boldsymbol{y} = \boldsymbol{f}$. Most GP inference efforts with non-Gaussian likelihoods $p(\boldsymbol{y}|\boldsymbol{f})$ are therefore focused on inferring the latent function $\boldsymbol{f}$ at the training points

$$p(\boldsymbol{f}|\boldsymbol{y},\boldsymbol{\theta}) \propto p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\theta})p(\boldsymbol{f}|\boldsymbol{\theta})\,. \tag{2.33}$$

[1]See, for example, `http://videolectures.net/mlss09uk_murray_mcmc/`

If $p(\boldsymbol{f}|\boldsymbol{y})$ is Gaussian, then the integral in Equation (2.32) is analytic and Gaussian. It is popular to approximate $p(\boldsymbol{f}|\boldsymbol{y})$ as a Gaussian, using the Laplace approximation, a variational approximation, or expectation propagation (EP).[1] EP has been especially successful for Gaussian process classification (Rasmussen and Williams, 2006). One can alternatively sample from $p(\boldsymbol{f}|\boldsymbol{y})$, and then approximate Equation (2.32) with a simple Monte carlo sum. Elliptical slice sampling (Murray et al., 2010) is particularly effective at sampling from posterior distributions like $p(\boldsymbol{f}|\boldsymbol{y})$ in Eq. (2.33), with strongly correlated (highly non-diagonal) Gaussian priors. We use variational inference and elliptical slice sampling in chapter 3 on *Gaussian process regression networks*. Wilson and Ghahramani (2010a) discusses the Laplace approximation for general non-Gaussian likelihoods.

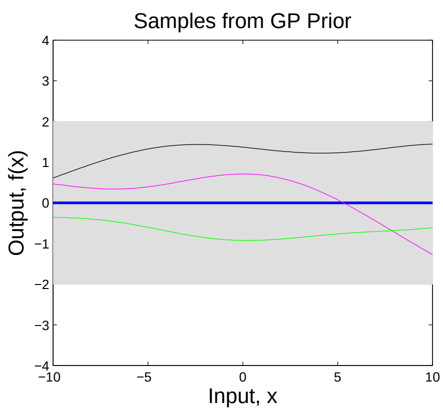
2.3.2 Learning and Model Selection

Since the properties of likely functions under a Gaussian process – smoothness, periodicity, rate of variability, etc. – are controlled by a covariance kernel and its hyperparameters, model selection and learning with Gaussian processes amounts to choosing a covariance kernel and learning its hyperparameters $\boldsymbol{\theta}$ from data.

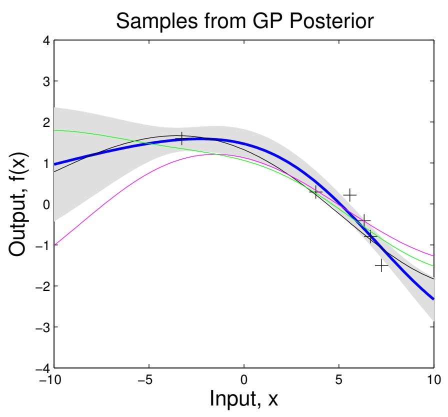
You may have found the Gaussian process sample functions in Figure 2.3 unsettling. While the datapoints are fit perfectly, the functions are too wiggly, and in places, the predictive uncertainty is unreasonably large. Indeed the *length-scale* hyperparameter of the SE kernel in Eq. (2.18) is too small, causing function values at nearby input points to be too uncorrelated. The length-scale specifies roughly on what scale a function will vary – what range of data is needed, for example, to make a good forecast. To produce Figure 2.4 we increased the length-scale of the SE kernel, and resampled functions from a GP. Now likely functions under the GP are too slowly varying – the support of the model is too concentrated on *simple* functions. We can see how the behaviour of a Gaussian process is extremely sensitive to even this single kernel hyperparameter – and thus how important it will be to properly determine this parameter. If we could automatically *learn* the

[1]The Laplace approximation, variational inference, and expectation propogation are discussed in a general setting in Bishop (2006). Rasmussen and Williams (2006) explicitly derive EP and the Laplace approximation for Gaussian process classification.

length-scale parameter from data – indeed if we could learn all hyperparameters $\boldsymbol{\theta}$ of a given covariance kernel, or ideally the entire functional form of the covariance function itself – we could perform automatic model selection over a massive space of models, and could discover interpretable properties of the data we are modelling.



(a)



(b)

Figure 2.4: Sample a) prior and b) posterior functions drawn from a GP with an SE kernel with a too large length-scale. The length-scale ℓ is too large, causing sample functions to vary too slowly. The GP mean function is shown in blue, and 95% of the probability mass about the mean function is shown with gray shade. The data are denoted by black markers.

In the kernel machines literature, particularly regarding support vector machines (Cortes and Vapnik, 1995), learning covariance kernel hyperparameters – aka model selection – is generally an unresolved problem, although there are many possibilities (Gönen and Alpaydın, 2011). Gaussian processes provide a formal and interpretable probabilistic framework for automatic model selection with kernel machines, which is especially suited to the expressive kernels we introduce in chapters 4 and 5.

Let us define a *model* $\mathcal{M}_i$ to mean a fully specified procedure for generating observations $\boldsymbol{y}$ – for example, a model is a function relating inputs to observations, including a noise model and any prior distributions. The posterior for a model $\mathcal{M}_i$

given observations $\boldsymbol{y}$ is given by

$$p(\mathcal{M}_i|\boldsymbol{y}) = \frac{p(\boldsymbol{y}|\mathcal{M}_i)p(\mathcal{M}_i)}{p(\boldsymbol{y})}\,. \tag{2.34}$$

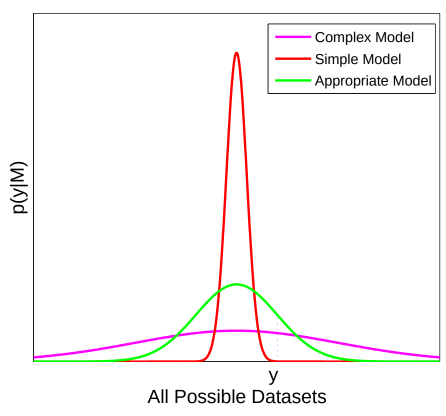
The normalizing constant $p(\boldsymbol{y})$ is $\sum_i p(\boldsymbol{y}|\mathcal{M}_i)p(\mathcal{M}_i)$. If the prior on models $p(\mathcal{M}_i)$ is flat, then the marginal likelihood (aka the *evidence*), $p(\boldsymbol{y}|\mathcal{M}_i)$, is directly proportional to $p(\mathcal{M}_i|\boldsymbol{y})$ in Eq. (2.34). The evidence is a probability distribution over all possible datasets, given a model specification $\mathcal{M}_i$. Therefore a model which gives high probability to certain datasets must give relatively little probability to other datasets, since $p(\boldsymbol{y}|\mathcal{M}_i)$ must normalize to 1 over all possible data sets $\boldsymbol{y}$. We define such a model to be a *simple* model.

We can write the marginal likelihood as

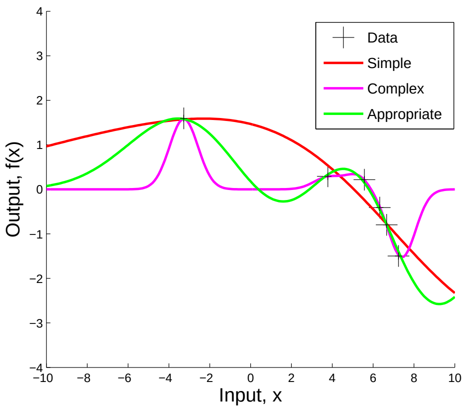
$$p(\boldsymbol{y}|\mathcal{M}_i) = \int p(\boldsymbol{y}|\boldsymbol{f}, \mathcal{M}_i)p(\boldsymbol{f})d\boldsymbol{f}\,, \tag{2.35}$$

where $\boldsymbol{f}$ are the parameters of the model (often parametrized as $\boldsymbol{w}$). Thus the evidence is the probability that randomly selected parameters from your model class would generate the data $\boldsymbol{y}$.

The first Gaussian process we encountered in this chapter, defined by Eqs. (2.1)-(2.2), is an example of a simple model: if we were to repeatedly sample a_0, a_1 from $\mathcal{N}(0, 1)$, $y(x) = a_0 + a_1 x$ could generate a small range of datasets with high probability. But since the evidence is a proper probability distribution, and must normalize, such a model could not support many other datasets, as depicted in Figure 2.5a. Conversely, a large Bayesian neural network, for example, might support almost any dataset with reasonable likelihood, but could not generate any particular dataset with high probability – such a model could be called *complex.* Thus for a particular dataset $\boldsymbol{y}$, the model with highest evidence will tend to be the simplest model that can explain the data. This tendency for the marginal likelihood to favour the simplest models that explain the data is sometimes called *automatic Occam's razor*, and it is explained pictorially in Figure 2.5a (MacKay, 1992a; Rasmussen and Ghahramani, 2001). We emphasize that complexity, in the sense of Figure 2.5a, does not have much to do with the number of parameters



(a)



(b)

Figure 2.5: Bayesian Occam's Razor applied to Gaussian Processes. a) The marginal likelihood (evidence) is on the vertical axis, and possible data sets are on the horizontal. Models of three different complexities are shown. The simple model can only explain a small number of data sets, but it explains them well: they have a high likelihood. On the other hand, the complicated model can explain a large number of data sets, but somewhat poorly. For a given data set, the model that maximizes the marginal likelihood (having fully integrated away the Gaussian process) will typically have an appropriate level of complexity. b) Posterior mean functions of a Gaussian process with SE kernel and too short, too large, and appropriate length-scales determined through marginal likelihood optimization.

in a model per se, but rather, the types of functions that are likely under a given model.

We have seen in Figures 2.4 that sample functions from a Gaussian process with a large length-scale are not flexible: the Gaussian process can support a small range of datasets with high probability, but has little support for most datasets. Conversely, sample functions from a GP with a short length-scale, e.g., Figure 2.3, are highly complex. If we were to marginalise away the latent Gaussian process $f(x)$, then we could use the resulting marginal likelihood to automatically calibrate model complexity.

The marginal likelihood (evidence) or probability density of the data $\boldsymbol{y}$, given hyperparameters $\boldsymbol{\theta}$ and inputs X, is

$$p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \int p(\boldsymbol{y}|\boldsymbol{f}, X)p(\boldsymbol{f}|\boldsymbol{\theta}, X)d\boldsymbol{f} \,. \tag{2.36}$$

If the likelihood of the data $p(\boldsymbol{y}|\boldsymbol{f}, X)$ is Gaussian, such that $p(y(x)|f(x)) = \mathcal{N}(y(x); f(x), \sigma^2)$, then this integral in Eq. (2.36) can be exactly computed such that

$$\log p(\boldsymbol{y}|\boldsymbol{\theta}, X) = \overbrace{-\frac{1}{2}\boldsymbol{y}^\top(K_{\boldsymbol{\theta}} + \sigma^2 I)^{-1}\boldsymbol{y}}^{\text{model fit}} - \overbrace{\frac{1}{2}\log|K_{\boldsymbol{\theta}} + \sigma^2 I|}^{\text{complexity penalty}} - \frac{N}{2}\log(2\pi)\,, \tag{2.37}$$

as in Eq. (2.23), where K is an $N \times N$ covariance matrix for N data points $\boldsymbol{y}$, with $K_{ij} = k(x_i, x_j|\boldsymbol{\theta})$.[1] The marginal likelihood in Eq. (2.37) pleasingly compartmentalises into model fit and model complexity terms, as expected from the discussion surrounding Figure 2.5. The model fit term is the only term that depends on the data. The complexity penalty $\log|K_{\boldsymbol{\theta}} + \sigma^2 I|$ is sometimes called *Occam's term* (Rasmussen and Williams, 2006), and is the volume of the ellipsoid defined by $\boldsymbol{z}^\top[K + \sigma^2 I]^{-1}\boldsymbol{z}^\top \leq 1$ for all $\boldsymbol{z} \in \mathbb{R}^N$. The more datasets $\boldsymbol{z}$ that can be accommodated by the model, the larger $\log|K + \sigma^2 I|$. Indeed both the model fit and model complexity terms in Eq. (2.37) grow monotonically with decreases in the length-scale hyperparameter.

The complexity penalty in Eq. (2.37) is automatically calibrated and profoundly different from the penalties found in *regularised* or penalised likelihoods which *are* often equivalent to the objective function used in *maximum a posteriori* (MAP) optimization. There is no need to calibrate a penalty term, e.g. using cross-validation.[2]

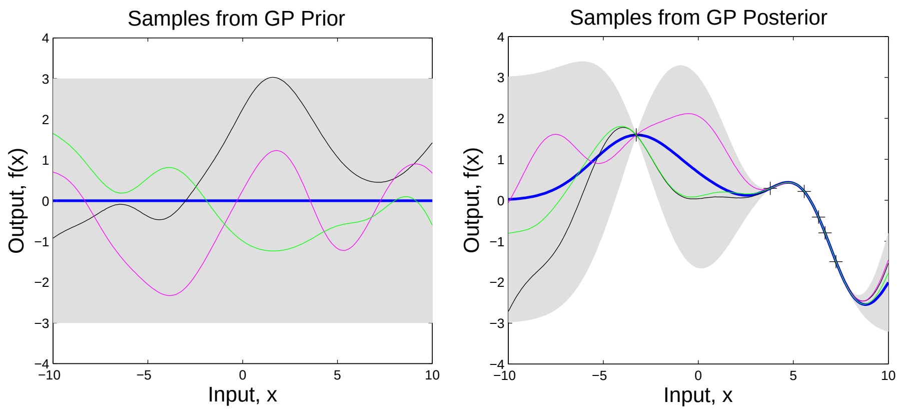
Now let us return to modelling the data in Figure 2.4. Suppose we choose an SE kernel, and optimise the marginal likelihood in Eq. (2.37) with respect to its hyperparameters $\boldsymbol{\theta}$ and σ^2, and sample from the posterior over Gaussian process functions. [3] We see in Figure 2.6 that after learning $\boldsymbol{\theta}$ through marginal likelihood optimization, the resulting functions intuitively appear to have an appropriate level of complexity – these functions are no longer too slowly or quickly varying. Figure

[1]I have overloaded notation. $K_{\boldsymbol{\theta}}$ simply means that the entries of K depend on $\boldsymbol{\theta}$. K_{ij} is the i^{th} row and j^{th} column of K.

[2]However, GP marginal likelihood optimization has a small bias towards under-fitting, which we discuss further in section 2.6.6.

[3]As discussed in section 2.3.1.1, the noise variance σ^2 can be incorporated into the kernel (when we have a Gaussian likelihood) and treated as one of the kernel hyperparameters $\boldsymbol{\theta}$, but for clarity we write these parameters separately.

2.5b shows the posterior mean functions from Figures 2.3, 2.4, and 2.6, for too simple, too complex, and appropriately complex models.



(a) (b)

Figure 2.6: Sample a) prior and b) posterior functions drawn from a GP with an SE kernel which has a length-scale that has been learned from the data. The length-scale ℓ has been learned from marginal likelihood optimization. Sample functions now appear to have an appropriate level of complexity. The GP mean function is shown in blue, and 95% of the probability mass about the mean function is shown with gray shade. The data are denoted by black markers.

In general, we ought not to only make predictions with the model that maximizes the evidence. Rather, we should perform a weighted average – Bayesian model averaging – over as large a space of models as is practically possible (Neal, 1996). The marginal likelihood could be multimodal as a function of parameters, and optimization of any likelihood can still lead to overfitting. Informally, one person's *marginal* likelihood is another person's likelihood: any likelihood can be derived by defining a hierarchical model and then marginalizing some parameters. Moreover, Figure 2.5a is only an intuition to explain the complexity penalty in the marginal likelihood. In section 1.2 of the introduction we considered a more precise and general definition of complexity, as part of a more general discussion on model selection.

Nonetheless, it is still remarkable that we can express the probability density of the data $\boldsymbol{y}$ only as a function of a few parameters (e.g. length-scale, and noise and signal variance), in such a flexible model as a Gaussian process with a squared exponential kernel. And, typically, optimizing the marginal likelihood with respect to only a small number of Gaussian process kernel hyperparameters $\boldsymbol{\theta}$ will not profoundly differ from a fully Bayesian treatment of hyperparameters. The predictive mean will be similar, and the predictive variance will tend to be somewhat too small (overconfident) when conditioning on optimized parameters $\boldsymbol{\theta}$, since this procedure erroneously assumes $\boldsymbol{\theta}$ are perfectly determined by the data.

A fully Bayesian treatment of Gaussian process models would integrate away kernel hyperparameters when making predictions:

$$p(\boldsymbol{f}_*|X_*, X, \boldsymbol{y}) = \int p(\boldsymbol{f}_*|X_*, X, \boldsymbol{y}, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\boldsymbol{y}) d\boldsymbol{\theta} \tag{2.38}$$

where the predictive distribution $p(\boldsymbol{f}_*|X_*, X, \boldsymbol{y}, \boldsymbol{\theta})$ is given by Eq. (2.24).

The posterior over hyperparameters $\boldsymbol{\theta}$ is given by

$$p(\boldsymbol{\theta}|\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{\theta}) p(\boldsymbol{\theta}) \,. \tag{2.39}$$

If the marginal likelihood $p(\boldsymbol{y}|\boldsymbol{\theta})$ is analytically tractable, as in Equation (2.37), then sampling from the posterior over hyperparameters $p(\boldsymbol{\theta}|\boldsymbol{y})$ is relatively straightforward. For example, one might place independent inverse Gamma priors over all parameters $\boldsymbol{\theta}$, and then sample from the posterior $p(\boldsymbol{\theta}|\boldsymbol{y})$ using Hamiltonian Monte Carlo (Duane et al., 1987; Neal, 2010; Rasmussen, 1996) or slice sampling (Murray and Adams, 2010; Neal, 2003). Note that after integrating away random hyperparameters, the resulting distribution over functions is no longer a Gaussian process. For example, if we scale a kernel k with a signal variance hyperparameter a^2, such that $k \to a^2 k$, and place an inverse Gamma prior on a^2, then the resulting process over functions after integrating away a^2 is a *Student-t process*, meaning that any collection of function values has a joint t distribution. Shah et al. (2014) discuss t processes in detail.

If the marginal likelihood is not analytically tractable, then hyperparameter estimation is generally difficult. Perhaps the most successful approach has been to approximate the marginal likelihood using, for example, Laplace, EP, or variational approximations, and then optimize the approximate marginal likelihood with respect to hyperparameters. This approximate marginal likelihood could also be used to sample hyperparameters in analogy with Eq. (2.39).

The reason a *marginal* likelihood is so important for inferring kernel hyperparameters, is because it gives us the likelihood of the data conditioned only on those hyperparameters. If we cannot (at least approximately) marginalise away the Gaussian process $f(x)$, then we have to deal with the extremely strong dependencies between the GP and its kernel hyperparameters. Indeed, in Figures 2.3, 2.4, 2.5, and 2.6, we see how profoundly the length-scale hyperparameter affects sample functions under a GP.

To sample from $p(\boldsymbol{\theta}|\boldsymbol{y})$ without an analytic marginal likelihood, one can alternately sample from

$$p(\boldsymbol{f}|\boldsymbol{\theta},\boldsymbol{y}) \propto p(\boldsymbol{y}|\boldsymbol{f},\boldsymbol{\theta})p(\boldsymbol{f}|\boldsymbol{\theta})\,, \tag{2.40}$$

$$p(\boldsymbol{\theta}|\boldsymbol{f}) \propto p(\boldsymbol{f}|\boldsymbol{\theta})p(\boldsymbol{\theta})\,, \tag{2.41}$$

in a Gibbs sampling scheme (Wilson and Ghahramani, 2010b, 2011), which converges to samples from the joint distribution $p(\boldsymbol{f},\boldsymbol{\theta}|\boldsymbol{y})$. The posterior over Gaussian process function values in Eq. (2.40) can often be efficiently sampled using elliptical slice sampling (Murray et al., 2010). Wilson and Ghahramani (2010b, 2011) have used slice sampling to sample from Eq. (2.41). This sampling scheme is simple to implement in general Gaussian process settings, but is computationally costly, since each new proposal for $\boldsymbol{\theta}$ naively requires constructing an $N \times N$ covariance matrix K, and then an $\mathcal{O}(N^3)$ decomposition of K. Moreover, the strong dependencies between $\boldsymbol{f}$ and $\boldsymbol{\theta}$ can lead to poor mixing. Murray and Adams (2010) have discussed sampling kernel hyperparameters in cases where a Gaussian process marginal likelihood is impossible, and have proposed solutions which appear to work nicely in special cases – for example, when a model has a likelihood that factorizes into a product, and each term in the product contains only a single Gaussian process function value. In intractable models which use multiple Gaussian processes, efficient sampling of hyperparameters is still an open problem. A robust and efficient method for estimating kernel hyperparameters in such models involves a variational Bayes approximation of the marginal likelihood, e.g. Wilson et al. (2012).

2.4 Covariance Kernels

At the heart of every Gaussian process model – *controlling all the modelling power* – is a covariance kernel. The kernel k directly specifies the covariance between a pair of random function values at a pair of input points: $k(x, x') = \text{cov}(f(x), f(x'))$. A covariance kernel therefore encodes inductive biases – what sorts of solution functions we expect. By choosing a covariance function, we choose whether the solution functions are periodic, smooth, linear, polynomial, etc. In other words, we hardly need to modify our algorithm to radically change the model – we just change the covariance function. Moreover, by estimating or inferring distributions over hyperparameters of a kernel, we can also discover interpretable properties of the data: periodicity, rate of variation, smoothness, etc.

We have so far used the terms *kernel*, *covariance kernel*, and *covariance function* interchangeably. In general, a *kernel* is a function that maps any pair of inputs into $\mathbb{R}$. The covariance function of a Gaussian process is an example of a kernel. For $k(x, x')$ to be a valid covariance function of a Gaussian process, any matrix K with elements $K_{ij} = k(x_i, x_j)$ must be positive semi-definite ($\boldsymbol{z}^\top K \boldsymbol{z} \geq 0$ for all $\boldsymbol{z} \in \mathbb{R}^N$); this requirement follows since the covariance matrix in a Gaussian distribution must be positive semi-definite. The positive semi-definite requirement is equivalent to requiring that the covariance function correspond to an inner product in some (potentially infinite) basis (feature) space (Bishop, 2006), although the positive semidefinite requirement is often more easily checked directly. We saw in section 2.2 that the popular linear weight-space model,

$$f(x) = \boldsymbol{w}^\top \boldsymbol{\phi}(x) \boldsymbol{w} \sim \mathcal{N}(0, \Sigma_w) \tag{2.42}$$

corresponds to a Gaussian process with covariance kernel

$$k(x, x') = \boldsymbol{\phi}(x)^\top \Sigma_w \boldsymbol{\phi}(x') \,. \tag{2.43}$$

We have also already accumulated some experience with the popular squared exponential (SE) kernel

$$k_{\text{SE}}(x, x') = a^2 \exp(-0.5||x - x'||^2/\ell^2) \,. \tag{2.44}$$

Below we list some simple techniques, reproduced from Bishop (2006) and MacKay (1998), for constructing new valid covariance functions, from existing covariance functions.

Suppose $k_1(x,x')$ and $k_2(x,x')$ are valid. Then the following covariance functions are also valid:

$$k(x,x') = g(x)k_1(x,x')g(x') \tag{2.45}$$

$$k(x,x') = q(k_1(x,x')) \tag{2.46}$$

$$k(x,x') = \exp(k_1(x,x')) \tag{2.47}$$

$$k(x,x') = k_1(x,x') + k_2(x,x') \tag{2.48}$$

$$k(x,x') = k_1(x,x')k_2(x,x') \tag{2.49}$$

$$k(x,x') = k_3(\boldsymbol{\phi}(x), \boldsymbol{\phi}(x')) \tag{2.50}$$

$$k(x,x') = x^\top A x' \tag{2.51}$$

$$k(x,x') = k_a(x_a,x'_a) + k_b(x_b,x'_b) \tag{2.52}$$

$$k(x,x') = k_a(x_a,x'_a)k_b(x_b,x'_b) \tag{2.53}$$

where g is any function, q is a polynomial with nonnegative coefficients, $\boldsymbol{\phi}(x)$ is a function from x to $\mathbb{R}^M$, k_3 is a valid covariance function in $\mathbb{R}^M$, A is a symmetric positive definite matrix, x_a and x_b are not necessarily disjoint variables with $x = (x_a, x_b)^\top$, and k_a and k_b are valid covariance functions in their respective spaces.

We will now proceed to describe the dot product, squared exponential, rational quadratic, neural network, Gibbs, Matérn, and periodic kernels in some detail, after introducing the concept of *stationarity*. At the end of this section we summarize some covariance functions in Table 2.1, including the new spectral mixture (SM) covariance function we introduce in chapter 4.

2.4.1 Stationary Kernels

A kernel is *stationary* if it is invariant to translations of the inputs: e.g., if the kernel is a function of $\tau = x - x'$, for any pair of inputs x and x'. Roughly speaking, the properties of a draw from a stationary process are similar at all x locations.

Isotropic kernels (*distance kernels*) are examples of stationary kernels. A kernel is *isotropic* if it is a function of $||x - x'||$.

The assumption of stationarity provides a useful inductive bias, and indeed most popular covariance kernels, including the squared exponential, rational quadratic, and Matérn kernels, to be discussed in the next sections, are not only stationary, but isotropic. Indeed it is difficult to learn anything about the covariance function of a stochastic process from a single realisation – e.g., to extract any information from the data – if we assume that the kernel is any positive definite function with equal probability, because such an assumption provides an unrealistic inductive bias.

Any stationary kernel (aka covariance function) can be expressed as an integral using Bochner's theorem (Bochner, 1959; Stein, 1999):

Theorem 2.4.1. *(Bochner) A complex-valued function k on $\mathbb{R}^P$ is the covariance function of a weakly stationary mean square continuous complex-valued random process on $\mathbb{R}^P$ if and only if it can be represented as*

$$k(\tau) = \int_{\mathbb{R}^P} e^{2\pi i s^\top \tau} \psi(\mathrm{d}s) \,, \tag{2.54}$$

where ψ is a positive finite measure.

If ψ has a density $S(s)$, then S is called the *spectral density* or *power spectrum* of k, and k and S are Fourier duals (Chatfield, 1989):

$$k(\tau) = \int S(s) e^{2\pi i s^\top \tau} ds \,, \tag{2.55}$$

$$S(s) = \int k(\tau) e^{-2\pi i s^\top \tau} d\tau \,. \tag{2.56}$$

In other words, a spectral density entirely determines the properties of a stationary kernel. And often spectral densities are more interpretable than kernels. If we Fourier transform a stationary kernel, the resulting spectral density shows the distribution of support for various frequencies. A heavy tailed spectral density has relatively large support for high frequencies. A uniform density over the spectrum corresponds to white noise (equivalent to an SE kernel with a length-scale $\ell \to 0$). Therefore draws from a process with a heavy tailed spectral density (such as an

Ornstein-Uhlenbeck process 2.4.8) tend to appear more erratic (containing higher frequencies, and behaving more like white noise) than draws from a process with spectral density that concentrates its support on low frequency functions. Indeed, as we will see, one can gain insights into kernels by considering their spectral densities.

For data on a regular 1D input grid of N points, we define the *empirical spectral density* $\hat{S}(s)$ as follows:

$$\hat{S}(s_m) = \frac{|\tilde{y}(s_m)|^2}{N}, \quad m = 0, \dots, N-1, \tag{2.57}$$

where $\tilde{y}(s_m)$ is the m^{th} element of the discrete Fourier transform (DFT) of the data vector $\boldsymbol{y}$. The empirical spectral density is defined for the frequencies $s_m = 0, f_s/N, 2f_s/N, \dots, f_s/2$, where f_s is the sampling rate of the data. Past the *Nyquist* frequency of $0.5f_s$, signals are aliased back to lower frequencies. That is, on a regularly spaced grid, samples from a function with a frequency $0.5f_s + \delta$ could be reproduced exactly by functions with frequencies $0.5f_s - \delta, 0.5f_s - \delta + f_s, 0.5f_s + \delta + fs, \dots$. It is common practice to filter out all but the lowest frequency alias. A derivation of the empirical spectrum can be found in section 2.6.3.1.

2.4.2 Dot Product Kernel

In section 2.2, we considered linear functions $f(x) = ax + b$, where $a \sim \mathcal{N}(0, \alpha)$ and $b \sim \mathcal{N}(0, \beta)$. As we have seen, it is easy to show that this random function is a Gaussian process with mean and covariance functions $m(x) = 0$ and $k(x, x') = \alpha^2 xx' + \beta$. In other words, if we draw from a GP using this mean and covariance function, we will be fitting our data with linear functions. This $k(x, x')$ is in the class of *dot product covariance functions*, which are non-stationary. We can generalize to the polynomial kernel, $k(x, x') = (x \cdot x' + \sigma_0^2)^p$. A GP with this kernel is exactly equivalent to using polynomial basis functions in a linear regression model. For example, if $p = 2$, $\sigma_0^2 = 0$ and the dimension of x is 2, then $k(x, x') = \boldsymbol{\phi}(x) \cdot \boldsymbol{\phi}(x)$ and $\boldsymbol{\phi}(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$. These are valid kernels since they have been derived as inner products of basis functions.

2.4.3 Squared Exponential Kernel

The squared exponential (SE) kernel, also sometimes called the Gaussian, radial basis function (RBF), or exponentiated quadratic kernel, is "probably the most widely-used kernel within the kernel machines field" (Rasmussen and Williams, 2006).

To derive the SE kernel[1], we start with the weight space model

$$f(x) = \sum_{i=1}^{J} w_i \phi_i(x)\,, \tag{2.58}$$

$$w_i \sim \mathcal{N}\left(0, \frac{\sigma^2}{J}\right)\,, \tag{2.59}$$

$$\phi_i(x) = \exp\left(-\frac{(x-c_i)^2}{2\ell^2}\right)\,. \tag{2.60}$$

Equations (2.58)-(2.61) define a radial basis function regression model, with radial basis functions centred at the points c_i.

From Eq. (2.43), the kernel of this Gaussian process is

$$k(x,x') = \frac{\sigma^2}{J}\sum_{i=1}^{J}\phi_i(x)\phi_i(x')\,. \tag{2.61}$$

Thus the entire radial basis weight space model of Eq. (2.58)-(2.60) is encapsulated as a distribution over functions with covariance kernel in Eq. (2.61). If we let $c_{i+1} - c_i = \Delta c = \frac{1}{J}$, then we recognize Eq. (2.61) as a Riemann sum:

$$k(x,x') = \lim_{J\to\infty}\frac{\sigma^2}{J}\sum_{i=1}^{J}\phi_i(x)\phi_i(x') = \int_{c_0}^{c_\infty}\phi_c(x)\phi_c(x')dc\,. \tag{2.62}$$

By setting $c_0 = -\infty$ and $c_\infty = \infty$, we spread the infinitely many basis functions across the whole real line, each a distance $\Delta c \to 0$ apart:

$$k(x,x') = \int_{-\infty}^{\infty}\exp(-\frac{x-c}{2\ell^2})\exp(-\frac{x'-c}{2\ell^2})dc \tag{2.63}$$

$$= \sqrt{\pi}\ell\sigma^2\exp(-\frac{(x-x')^2}{2(\sqrt{2}\ell)^2})\,. \tag{2.64}$$

[1]For notational simplicity, our derivation assumes the inputs $x \in \mathbb{R}^1$. Extension to inputs $x \in \mathbb{R}^P$ is straightforward.

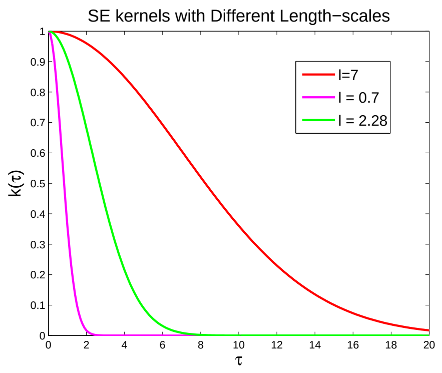


Figure 2.7: SE kernels with different length-scales, as a function of $\tau = x - x'$.

Therefore a Gaussian process with an SE kernel

$$k_{\mathrm{SE}}(x, x') = a^2 \exp(-0.5||x - x'||^2/\ell^2)\,, \tag{2.65}$$

is equivalent to a radial basis function model with infinitely many basis functions densely scattered across the whole real line, yet inference and predictions can be performed using a kernel representation, as in section 2.3, using finite computational resources!

Functions sampled from a Gaussian process with an SE kernel are infinitely differentiable. A Gaussian process with SE kernel is also a *universal approximator*: given enough data a GP with an SE kernel can approximate *any* function arbitrarily well (Ghosal and Roy, 2006; van der Vaart and van Zanten, 2009). Moreover, a GP with SE kernel has support for any continuous function to within an arbitrarily small ϵ band (Ghosal and Roy, 2006).

Figure 2.7 shows SE kernels with various length-scales as a function of $\tau = x - x' \in \mathbb{R}^1$. We have already seen sample functions from SE kernels with each of these length-scales in Figures 2.3, 2.4, and 2.6.

Taking the Fourier transform of the SE kernel, we find its spectral density for inputs $x \in \mathbb{R}^P$ is $S_{\mathrm{SE}}(s) = (2\pi\ell^2)^{P/2} \exp(-2\pi^2\ell^2 s^2)$. The spectral density thus places most of its support (aka power) on low frequencies.

2.4.4 Rational Quadratic Kernel

The squared exponential kernel assumes that the data are only varying at one particular length-scale. In reality, however, different mechanisms underlying the data could be varying on different scales. For example, Brownlees et al. (2009) found that the variance (volatility) on returns on equity indices have patterns that vary over different scales. Thus, to model the volatility of equity index returns, it might be sensible to use a sum of SE kernels, each with different length-scales learned from the data.

However, we may often be unsure about the scales over which data are varying – and indeed we may wish to account for infinitely many possible scales. The rational quadratic (RQ) kernel is a scale mixture (infinite sum) of squared exponential kernels with different length-scales.

Expressing the SE kernel as a function of $r = ||x - x'||$, we can write a general scale mixture of SE kernels as

$$k(r) = \int \exp(-\frac{r^2}{2l^2}) p(\ell) d\ell \,, \tag{2.66}$$

where $p(\ell)$ is a distribution over length-scales ℓ.

If we place a Gamma distribution on inverse squared length-scales, $\gamma = \ell^{-2}$, $g(\gamma|\alpha, \beta) \propto \gamma^{\alpha-1} \exp(-\alpha\gamma/\beta)$, with $\beta^{-1} = \ell'^2$, the RQ kernel is derived as

$$\int_0^\infty k_{\mathrm{SE}}(r|\gamma) g(\gamma|\alpha, \beta) d\gamma = (1 + \frac{r^2}{2\alpha\ell'^2})^{-\alpha} \,. \tag{2.67}$$

The RQ kernel is intended to model multi-scale data. Figure 2.8 shows the RQ kernel, and sample functions, for various settings of α. As $\alpha \to \infty$, the rational quadratic kernel converges to the SE kernel. One could derive other interesting covariance functions using different (non Gamma) density functions for $p(\ell)$.

2.4.5 Neural Network Kernel

The neural network kernel is perhaps most notable for catalyzing research on Gaussian processes within the machine learning community. Neal (1996) argued

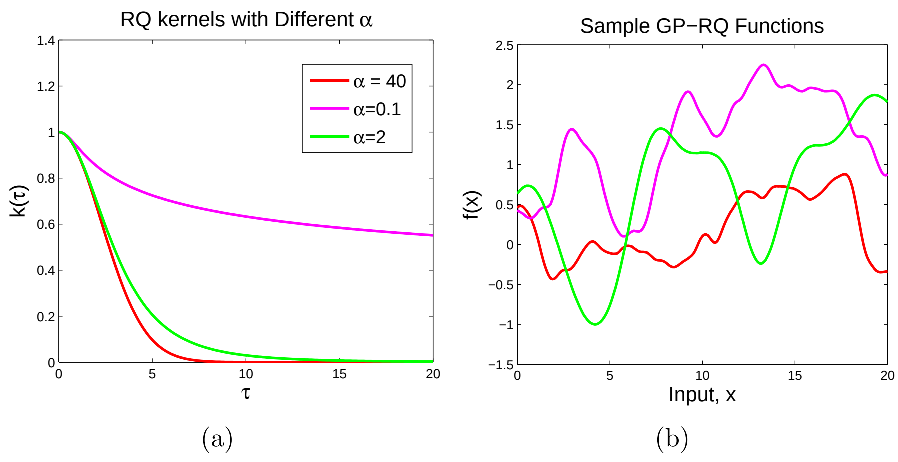


Figure 2.8: The rational quadratic (RQ) kernel. a) RQ kernels with three different settings of alpha, each with the 'appropriate' length-scale of 2.28 used in the Figures for the SE kernel. b) Sample functions from GPs with each of these kernels. Decreasing α significantly broadens the tails of the RQ kernel, increasing support for higher frequencies. Thus sample functions from an RQ kernel with a smaller α have less smooth (more erratic) appearance.

that since Bayesian models do not overfit, we ought to use expressive models that are more capable of describing sophisticated real world processes. No matter how sophisticated the model, there is likely some fresh detail in data that could be considered for improved modelling performance. Accordingly, Neal (1996) pursued the limits of large models, and showed that a Bayesian neural network becomes a Gaussian process with a *neural network kernel* as the number of units approaches infinity. This observation inspired Williams and Rasmussen (1996) to further explore Gaussian process models.

Following Neal (1996); Rasmussen and Williams (2006); Williams (1998), we consider a neural network with one hidden layer:

$$f(x) = b + \sum_{i=1}^{J} v_i h(x; \boldsymbol{u}_i) \,. \tag{2.68}$$

v_i are the hidden to output weights, h is any bounded hidden unit transfer function, $\boldsymbol{u}_i$ are the input to hidden weights, and J is the number of hidden units.

We let the bias b and hidden to output weights v_i have independent zero mean distributions with variances σ_b^2 and σ_v^2/J, and the weights for each hidden unit $\boldsymbol{u}_i$ have independent and identical distributions.

The first two moments of $f(x)$ in Eq. (2.68), collecting all weights together into the vector $\boldsymbol{w}$, are

$$\mathbb{E}_{\boldsymbol{w}}[f(x)] = 0\,, \tag{2.69}$$

$$\mathrm{cov}[f(x), f(x')] = \mathbb{E}_{\boldsymbol{w}}[f(x)f(x')] = \sigma_b^2 + \frac{1}{J}\sum_{i=1}^{J} \sigma_v^2 \mathbb{E}_{\boldsymbol{u}}[h_i(x;\boldsymbol{u}_i)h_i(x';\boldsymbol{u}_i)]\,, \tag{2.70}$$

$$= \sigma_b^2 + \sigma_v^2 \mathbb{E}_{\boldsymbol{u}}[h(x;\boldsymbol{u})h(x';\boldsymbol{u})]\,. \tag{2.71}$$

Eq. (2.71) follows from Eq. (2.70) since each of the $\boldsymbol{u}_i$ are identically distributed. The sum in Eq. (2.70) is over J i.i.d. random variables, and all moments are bounded. If b has a Gaussian distribution, the central theorem can be applied to show that as $J \to \infty$ any collection of function values $f(x_1), \ldots, f(x_N)$ has a joint Gaussian distribution, and thus the neural network in Eq. (2.70) becomes a Gaussian process with covariance function given by Eq. (2.71), with a rate of convergence of $J^{-0.5}$.

If we choose the transfer function as $h(x;\boldsymbol{u}) = \mathrm{erf}(u_0 + \sum_{j=1}^{P} u_j x_j)$, where $\mathrm{erf}(z) = \frac{2}{\sqrt{\pi}}\int_0^z e^{-t^2}dt$, and we choose $\boldsymbol{u} \sim \mathcal{N}(0, \Sigma)$, then we obtain (Williams, 1998) from Eq. (2.71)

$$k_{\mathrm{NN}}(x, x') = \frac{2}{\pi}\sin\left(\frac{2\tilde{x}^\top \Sigma \tilde{x}'}{\sqrt{(1 + 2\tilde{x}^\top \Sigma \tilde{x})(1 + 2\tilde{x}'^\top \Sigma \tilde{x}')}}\right)\,, \tag{2.72}$$

where $x \in \mathbb{R}^P$ and $\tilde{x} = (1, x^\top)^\top$.

As discussed in Rasmussen and Williams (2006), samples from a GP with this kernel are superpositions of the functions $\mathrm{erf}(u_0 + \boldsymbol{u}^\top x)$ – which is consistent with how draws tend to a constant value for large positive or negative x; indeed this is a non-stationary covariance function.

The squared exponential kernel, derived in section 2.4.3, can be re-derived using an infinite neural network with transfer function $h(x;\boldsymbol{u}) = \exp(-||x - \boldsymbol{u}||^2/\sigma_l^2)$ and $\boldsymbol{u} \sim \mathcal{N}(0, \sigma_u^2 I)$, which results in a generalised form of the SE kernel (Rasmussen and Williams, 2006).

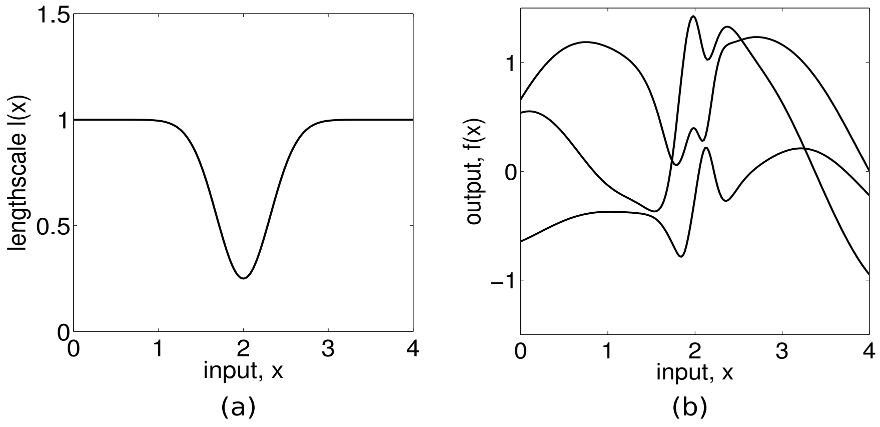


Figure 2.9: GP sample functions using a non-stationary Gibbs covariance function, with an input dependent length-scale. In a) is the length-scale function $l(x)$, and in b) are sample functions from a Gaussian process using this length-scale with the Gibbs covariance function (2.73). This figure is reproduced from Rasmussen and Williams (2006).

2.4.6 Gibbs Kernel

Suppose that we wanted a non-stationary covariance function that had an input dependent length-scale, $l(x)$. We cannot, for instance, simply replace l in the squared exponential covariance function of Eq. (2.65) with any $l(x)$ and have a valid covariance function. For $k(x, x')$ to be a valid covariance function, any matrix K with elements $k(x_n, x_m)$ must be positive semidefinite; this follows since the covariance matrix in a Gaussian distribution must be positive semidefinite. Gibbs (1997) derived a valid covariance function with an input dependent length-scale:

$$k(x, x') = \prod_{p=1}^{P} \Big(\frac{2l_p(x)l_p(x')}{l_p^2(x) + l_p^2(x')}\Big)^{1/2} \exp\Big(-\sum_{p=1}^{P} \frac{(x_p - x'_p)^2}{l_p^2(x) + l_p^2(x')}\Big)\,, \tag{2.73}$$

where x_p is the p^{th} component of x. In Figure 2.9b we see some Gaussian process sample functions drawn using the Gibbs covariance function (2.73) with the length-scale function in a).

2.4.7 Periodic Kernel

One technique for creating a valid non-stationary covariance function is to map the inputs through a non-linear function $\boldsymbol{v}(x)$, and then use a stationary covariance function in $\boldsymbol{v}$-space. MacKay (1998) uses this transformation in a different way to create a stationary periodic covariance function. He uses the transformation $\boldsymbol{v} = (\cos(x), \sin(x))$, and then applies the squared exponential covariance function (2.65) in $\boldsymbol{v}$-space to get

$$k(x, x') = \exp(-\frac{2\sin^2(\frac{x-x'}{2})}{l^2}) \,. \tag{2.74}$$

Although this kernel gives rise to periodic functions, notice that it is strictly positive – as with all stationary kernels presented in this section. We would intuitively expect periodic functions, like a sinusoid, to have negative covariances, since peaks are anticorrelated with troughs.

This kernel is mostly useful in combination with other covariance functions. For example, Rasmussen and Williams (2006) uses this kernel in combination with squared exponential (SE) and rational quadratic (RQ) kernels to model oscillatory CO_2 data with a rising trend.

2.4.8 Matérn Kernel

The Matérn kernel is likely the second most popular kernel, after the squared exponential. Stein (1999) argues that the smoothness assumption of the squared exponential kernel is unrealistic for modelling physical processes, and recommends the Matérn kernel as an alternative.

The Matérn kernel can be derived by modelling a spectral density $S(s)$ as a t-distribution, and then performing the integral in Eq. (4.3), which has an analytic solution. The heavy tails of a t-distribution in frequency space will give more importance ('power') to higher frequencies. Sample functions from a Matérn kernel are finitely differentiable.

The most general form of the Matérn kernel is

$$k_{\text{Matérn}}(x, x') = \frac{2^{1-\nu}}{\Gamma(\nu)}(\frac{\sqrt{2\nu}|x - x'|}{l})^{\nu} K_{\nu}(\frac{\sqrt{2\nu}|x - x'|}{l})\,, \tag{2.75}$$

where K_ν is a modified Bessel function (Abramowitz and Stegun, 1964).

In one dimension, and when $\nu + 1/2 = p$, for some natural number p, the corresponding GP is a continuous time AR(p) process. By setting $\nu = 1$, we obtain the *Ornstein-Uhlenbeck* (OU) kernel,

$$k(x, x') = \exp(-\frac{|x - x'|}{l})\,, \tag{2.76}$$

which is the covariance function of an Ornstein-Uhlenbeck process (Uhlenback and Ornstein, 1930), introduced to model the velocity of a particle undergoing Brownian motion.

Recently (Le et al., 2013), the Matérn kernel has been found to have superior performance to the squared exponential kernel on datasets with high dimensional inputs, e.g. $x \in \mathbb{R}^P$, $P \gg 1$. It has been speculated that this improved performance is because the Matérn kernel obviates a 'concentration of measure' effect in high dimensions. The squared exponential or (Gaussian) kernel can be derived from a Gaussian spectral density. In high dimensions we can imagine samples from this density function will be highly constrained to the surface of P dimensional ellipse – and thus all samples will have a similar characteristic scale. This problem is less severe with a high dimensional (heavy tailed) t-distribution for the spectral density. It will be interesting to further investigate the use of Matérn kernels in high dimensional input spaces.

2.4.9 Summary

We have seen how all sorts of models, even infinite order models, can be simply encapsulated by a Gaussian process with a mean function and covariance kernel. The kernel is indeed the heart of a Gaussian process, and it can be used to profoundly change a Gaussian process model, without needing to change inference and learning procedures. This thesis is concerned with developing expressive kernels

Table 2.1: Example covariance functions.

Covariance function	Expression	Stationary
Constant	a_0	Yes
Linear	$x \cdot x'$	No
Polynomial	$(x \cdot x' + a_0)^p$	No
Squared Exponential	$\exp(-\frac{\lvert x-x'\rvert^2}{2l^2})$	Yes
Matérn	$\frac{2^{1-\nu}}{\Gamma(\nu)}(\frac{\sqrt{2\nu}\lvert x-x'\rvert}{l})^{\nu} K_{\nu}(\frac{\sqrt{2\nu}\lvert x-x'\rvert}{l})$	Yes
Ornstein-Uhlenbeck	$\exp(-\frac{\lvert x-x'\rvert}{l})$	Yes
Rational Quadratic	$(1+\frac{\lvert x-x'\rvert^2}{2\alpha l^2})^{-\alpha}$	Yes
Periodic	$\exp(-\frac{2\sin^2(\frac{x-x'}{2})}{l^2})$	Yes
Gibbs	$\prod_{p=1}^{P}\big(\frac{2l_p(x)l_p(x')}{l_p^2(x)+l_p^2(x')}\big)^{1/2}\exp\big(-\sum_{p=1}^{P}\frac{(x_p-x'_p)^2}{l_p^2(x)+l_p^2(x')}\big)$	No
Spectral Mixture	$\sum_{q=1}^{Q} w_q \prod_{p=1}^{P}\exp\{-2\pi^2(x-x')_p^2 v_{qp}\}\cos(2\pi(x-x')_p\mu_{qp})$	Yes

to learn structure in data, and exploiting the existing structure in these kernels, for exact efficient inference and learning.

In Table 2.1 we summarize some covariance kernels, including the new spectral mixture kernel we introduce in chapter 4, for automatic pattern discovery and extrapolation.

One can use a combination of covariance functions to model data. For example, squared exponential and periodic covariance functions could be used together, e.g. through the identities (2.48) and (2.49), to capture both an increasing trend and periodicity.

2.5 The Mean Function

The mean function of a Gaussian process is often ignored. It is common practice to subtract the empirical mean from the data, and then assume a zero mean function. However, like the covariance function, the mean function is also a powerful way to encode assumptions (inductive biases) into a Gaussian process model, important for extracting useful information from a given dataset.

For example, an objection I have often heard to using Gaussian processes – and flexible Bayesian nonparametric models in general – is that they do not make use of any physical model we might have for the data. Surely – the argument runs – it is more sensible to write down the parametric physical model for a system (if one exists), and then estimate unknown parameters in that model, rather than use some flexible black box function approximator.

However, any parametric functional form (e.g. any physical model) is going to be wrong – often very wrong – and honestly reflecting uncertainty in our efforts to model data will yield better inferences and predictions. If we do have a parametric physical model for the data, we can always use this model as the mean function of a Gaussian process. We can then concentrate the support of the distribution over functions induced by a GP around the mean function, using a signal variance parameter in the kernel (e.g., the a^2 term in the SE kernel of Eq. (2.65)). How much we concentrate this support, as well as the unknown parameters in the physical model, can easily be learned from data. Any parameters of the mean function will become hyperparameters in a marginal likelihood, and can be learned in the same way as covariance kernel hyperparameters (learning techniques for hyperparameters are discussed in section 2.3.2).

In short, we can leverage the assumptions of any parametric function in a Gaussian process through a mean function, and still retain support for other possible solution functions. All solution functions will be concentrated around the mean function, e.g. a desired parametric model, to an extent suggested by the data, or hard-coded by the user. Thus, to faithfully represent our beliefs, one should *never* exclusively use a physically (or otherwise) motivated parametric model in favour of a Gaussian process: the Gaussian process can leverage the assumptions of a parametric model through a mean function and also reflect the belief that the parametric form of that model will not be entirely accurate.

2.6 The Future of Gaussian Processes

In this section, I give my views on the future of Gaussian process research – my intuitions about what will become mainstream in five years time, and what present limitations have so far held back Gaussian processes from widespread applications, and potential solutions, with references to some of my new work in this thesis.

2.6.1 Bayesian Nonparametric Gaussian Processes

Bayesian nonparametric Gaussian processes are not new, but given the recent movement towards finite basis function models for large datasets (Lázaro-Gredilla et al., 2010; Le et al., 2013; Rahimi and Recht, 2007; Williams and Seeger, 2001), discussed further in section 2.6.3, we ought not to overlook the importance of non-parametric Gaussian process representations in the future.

Parametric models are completely described by a finite set of parameters $\boldsymbol{w}$. These models therefore have fixed degrees of freedom. The capacity of a parametric model – the amount of information that the model can represent – "is bounded, even if the amount of observed data becomes unbounded" (Ghahramani, 2013). Conversely, non-parametric models have an infinite number of parameters – and the number of free parameters grows with the size of the data. In this sense, non-parametric models have the desirable property that they can extract more information when there is more information available. Indeed in the introductory chapter to this thesis, I argued that Bayesian nonparametric representations have many ideal properties. E.g., Bayesian nonparametric models are naturally suited to learning rich information from the data, and reflect the belief that the real world is highly sophisticated.

The *function space* view of Gaussian process regression, presented in section 2.2, complements nonparametric modelling. In a sense, the parameters of the function space model are the function values $\boldsymbol{f}$ (evaluated at training locations) themselves, and the number of function values describing a given dataset equals the number of points in that dataset. However, not all Gaussian processes are Bayesian non-parametric models: indeed we have shown how many basic parametric models are

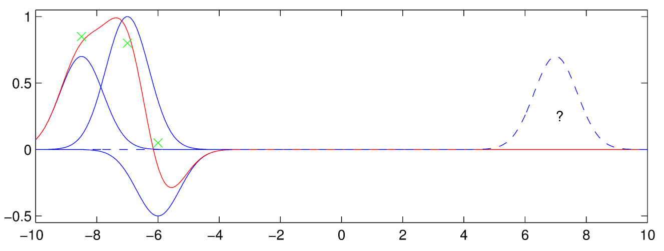


Figure 2.10: Finite radial basis function (RBF) regression versus Gaussian process regression with an SE kernel. Data points are shown with green crosses, basis functions for the finite RBF model are shown in blue, and predictions made using the RBF model are shown in red. With only finitely many basis functions, the RBF model is constrained to have a high confidence in its predictions far away from the data – even though we ought to have low confidence in this region. A Gaussian process model with a squared exponential kernel, however, is equivalent to RBF regression with infinitely many basis functions everywhere along the input domain, and can therefore properly represent prior uncertainty far away from the data. This image originally appeared in a talk by Carl Edward Rasmussen at the 2009 Cambridge Machine Learning Summer School: `http://videolectures.net/mlss09uk_rasmussen_gp/`

examples of Gaussian processes. For a Gaussian process to be non-parametric, each component function value $f(x_i)$ in any collection $\boldsymbol{f} = (f(x_1), \dots, f(x_N))^\top$ must be free to take any value, regardless of the other function values. Expressed differently, the conditional distribution $f(x_i)|\boldsymbol{f}_{-i}$, where $\boldsymbol{f}_{-i}$ is any collection of function values excluding $f(x_i)$, must be free to take any value in $\mathbb{R}$. For this freedom to be possible it is a necessary (but not sufficient) condition for the kernel of the Gaussian process to be derived from an infinite basis function expansion, such as the squared exponential kernel in section 2.4.3.

In an abstract sense, it is straightforward that more basis functions leads to more flexibility and is therefore desirable, given that over-fitting is not an issue with Bayesian inference. However, it is enlightening to explicitly compare the differences between infinite basis function models and their finite dimensional analogues.

Suppose we observe three data points, shown with green crosses in Figure 2.10. We can model these data by centering a Gaussian basis function (aka a radial basis function) on each data point. We can place a Gaussian prior distribution over the weights in this model, with zero mean, and perform fully Bayesian inference to make predictions; this model with finite basis functions is indeed a Gaussian process. Suppose now that we wish to make a prediction far away from the data, e.g., at $x = 7$. The radial basis function model will predict a value near 0 with high confidence (low variance). However, we cannot be confident about what is happening far away from the data – so something is wrong with this finite basis function model.

We can also model these datapoints using a Gaussian process with a squared exponential kernel, which, as derived in section 2.4.3, corresponds to using an infinite number of densely dispersed radial basis functions. The mean prediction of the Gaussian process with an SE kernel at the test location $x = 7$ will still be zero, but now the predictive variance is non-zero, as we would expect. The GP with SE kernel places basis functions around the test location, which will not be too influenced by the far away data. The GP with SE kernel has this nonparametric flexibility for free: the model is equivalent to placing an infinite number of radial basis functions across the whole real line, but predictions can be made with finite computation and memory.

In the future, where it appears "big data" will be of great interest, we must not forget why we moved away from finite basis function models in the first place. The infinite models provide incredible flexibility, and more faithfully reflect our prior beliefs. With more training data available in the future, we have a greater opportunity to learn sophisticated structure in data and therefore it will only be more important to have expressive models. The need for expressive models is a theme across the next two sections on the future of Gaussian processes.

In chapter 5, we show how nonparametric Gaussian process representations – and their particular ability to capture more information when more data are available – are one of many critical ingredients for successful pattern discovery and extrapolation.

2.6.2 Did We Throw the Baby Out with the Bathwater? (Expressive New Kernels)

In machine learning, Gaussian processes developed out of neural networks research, partly in response to the unanswered questions concerning neural network architectures, activation functions, etc. (e.g., Bishop (1995); Rasmussen and Williams (2006)) .

Gaussian processes, by contrast, are flexible *and* interpretable and manageable, with easy model specification through covariance kernels, and a principled framework for learning kernel hyperparameters.

However, neural networks became popular because they could automatically discover interesting hidden representations in data. They were part of a machine learning dream of developing intelligent agents that would automatically learn and make decisions. Gaussian processes, by contrast, are simply smoothing devices, if used with popular kernels, such as the squared exponential (SE) kernel, which cannot discover sophisticated structure in data and extrapolate. This observation prompted MacKay (1998) to ask whether we had "thrown the baby out with the bathwater" in treating Gaussian processes as a potential replacement for neural networks.

The resolution to this problem – more expressive covariance kernels – is the main subject of this thesis. In the next chapter, on Gaussian process regression networks, we introduce a highly flexible Gaussian process model which can discover interpretable features in data, through input dependent mixtures of 'base' kernels. In chapters 4 and 5 we introduce expressive closed form kernels, which form a highly effective basis for all stationary covariance functions. Overall, expressive kernels allow us to extract additional information from the data – to discover scientifically interesting properties in the data, and profoundly improve predictive accuracy.

In general, expressive covariance functions are "an important area of future developments for GP models" (Rasmussen and Williams, 2006). Indeed GPs are particularly suited to expressive covariance functions, since GPs provide a powerful framework for hyperparameter learning – a framework which has not yet been fully exploited with the simple covariance functions which are popular today.

At present, research on expressive kernels is in its infancy. We cannot even typically model negative covariances with kernel functions popular in the machine learning community.[1] Yet negative covariances are such a basic property of so many datasets – even linear trends will have long range negative covariances. We discuss modelling negative covariances further in section 4.3.3.

2.6.3 Exact Efficient Inference

With increasingly large datasets, there has been a growing movement towards improving the scalability of Gaussian process based kernel machines, which are often seen as having prohibitive computational and memory requirements.

Gaussian process inference and learning requires evaluating $(K + \sigma^2 I)^{-1}\boldsymbol{y}$ and $\log|K + \sigma^2 I|$, for an $N \times N$ covariance matrix K, a vector of N training instances $\boldsymbol{y}$, and noise variance σ^2, as in Equations (2.27) and (2.37), respectively. For this purpose, it is standard practice to take the Cholesky decomposition of $(K + \sigma^2 I)$ which requires $\mathcal{O}(N^3)$ computations and $\mathcal{O}(N^2)$ storage.

Methods to reduce these computational limitations often involve simplifying assumptions, such as finite basis function expansions (Lázaro-Gredilla et al., 2010; Le et al., 2013; Rahimi and Recht, 2007; Williams and Seeger, 2001), or sparse approximations using pseudo inputs (Csató and Opper, 2001; Quiñonero-Candela and Rasmussen, 2005; Rasmussen and Williams, 2006; Seeger et al., 2003; Snelson and Ghahramani, 2006), which scale as $\mathcal{O}(M^2 N)$, where $M \ll N$ is a fixed number of pseudo (aka inducing) inputs or basis functions.[2] The number of required pseudo inputs or basis functions for good performance will generally increase with the size of the dataset. While these methods are successful with simple smoothing

[1]Basic discrete time autoregressive Gaussian processes, however, often express negative correlations. See, for example, section 4.3.3 or appendix A.

[2]Hensman et al. (2013) recently developed an inducing input approach which scales as $\mathcal{O}(M^3)$, for smoothing and interpolation on large (e.g., $N \approx 10^6$) datasets.

kernels, they can be extremely prohibitive when combined with more expressive models (e.g., section 2.6.2 and chapter 5) – sacrificing the structure, or information in the data, which makes these expressive models useful in the first place.

However, there is often significant structure in the covariance matrix K, which can be exploited for fast and *exact* learning and inference procedures. Indeed, a covariance kernel must be structured to reflect the inductive biases in any given model. I believe methods that exploit existing structure in kernels for exact inference and learning will play an important role in the future of Gaussian process research and applications. In section 5 we show how expressive kernels, nonparametric representations, and scalable and exact inference and learning are all critical for large scale multidimensional pattern extrapolation with Gaussian processes.

Here we will briefly describe two complementary approaches which exploit kernel structure for exact scalable inference. Both of these methods allow us to efficiently determine the eigenvalues of a covariance matrix K, which in turn enables us to efficiently compute $(K+\sigma^2 I)^{-1}\boldsymbol{y}$ and $\log|K+\sigma^2 I|$, for fast inference and learning.

2.6.3.1 Toeplitz Structure

Given N training instances on a regularly spaced 1D input grid, a stationary kernel k (section 2.4.1) will give rise to a Toeplitz covariance matrix K, meaning that every diagonal of K is the same (note also K is a covariance matrix, so is symmetric as well as Toeplitz)[1]

$$K = \begin{bmatrix} k_0 & k_1 & \dots & k_{N-2} & k_{N-1} \\ k_1 & k_0 & \dots & k_{N_3} & k_{N-2} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ k_{N-2} & k_{N-3} & \dots & k_0 & k_1 \\ k_{N-1} & k_{N-2} & \dots & k_1 & k_0 \end{bmatrix} . \tag{2.77}$$

[1]Since k is stationary, we write $k(x,x') = k(x-x') = k_\tau$.

One can embed this $N \times N$ Toeplitz matrix K into a $2(N+1) \times 2(N+1)$ circulant matrix C (every column is shifted a position from the next):

$$C = \left[\begin{array}{ccccc|ccccc}
k_0 & k_1 & \dots & k_{N-2} & k_{N-1} & k_{N-2} & k_{N-3} & \dots & k_2 & k_1 \\
k_1 & k_0 & \dots & k_{N-3} & k_{N-2} & k_{N-1} & k_{N-2} & \dots & k_3 & k_2 \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
k_{N-2} & k_{N-3} & \dots & k_0 & k_1 & k_2 & k_3 & \dots & k_{N-2} & k_{N-1} \\
k_{N-1} & k_{N-2} & \dots & k_1 & k_0 & k_1 & k_2 & \dots & k_{N-3} & k_{N-2} \\
\hline
k_{N-2} & k_{N-1} & \dots & k_2 & k_1 & k_0 & k_1 & \dots & k_{N-4} & k_{N-3} \\
k_{N-3} & k_{N-2} & \dots & k_3 & k_2 & k_1 & k_0 & \dots & k_{N-3} & k_{N-2} \\
\vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\
k_2 & k_3 & \dots & k_{N-2} & k_{N-3} & k_{N-4} & k_{N-3} & \dots & k_0 & k_1 \\
k_1 & k_2 & \dots & k_{N-1} & k_{N-2} & k_{N-3} & k_{N-2} & \dots & k_1 & k_0
\end{array}\right]$$

$$= \begin{bmatrix} K & S \\ S^\top & A \end{bmatrix} . \tag{2.78}$$

Notice that all of the information in C is contained in its first column c, and that entry $C_{t,j} = c_{(j-t) \bmod N}$.

In finding the eigenvectors and eigenvalues of C, we can formulate an efficient procedure for solving $K^{-1}\boldsymbol{y}$ and $|C|$. An eigenvector $\boldsymbol{u}$ and eigenvalue λ of C satisfy

$$C\boldsymbol{u} = \lambda\boldsymbol{u} \, , \tag{2.79}$$

which is equivalent to the N difference equations

$$\sum_{t=0}^{m-1} c_{N-m+t} u_t + \sum_{t=m}^{N-1} c_{t-m} u_t = \lambda u_m \, , \quad m = 0, \dots, N-1 \, , \tag{2.80}$$

where u_t is the t^{th} element of $\boldsymbol{u}$. Eq. (2.80) can be re-written as

$$\sum_{t=0}^{N-1-m} c_t u_{t+m} + \sum_{t=N-m}^{N-1} c_t u_{t-(N-m)} = \lambda u_m \, . \tag{2.81}$$

As with differential equations, a common strategy in solving difference equations is to guess and then verify the form of the solution. Following Gray (2006), a reasonable guess for a system which is linear with constant coefficients is $u_t = \gamma^t$.

Substituting $u_t = \gamma^t$ into Eq. (2.81), we find

$$\sum_{t=0}^{N-1-m} c_t \gamma^t + \gamma^{-N} \sum_{t=N-m}^{N-1} c_t \gamma^t = \lambda \,. \tag{2.82}$$

For $\gamma^{-N} = 1$, the solution eigenvalues and eigenvectors are

$$\lambda = \sum_{t=0}^{N-1} c_t \gamma^t \,, \tag{2.83}$$

$$\boldsymbol{u} = N^{-1/2}(1, \gamma, \gamma^2, \ldots, \gamma^{N-1})^\top \,. \tag{2.84}$$

Solving $\gamma^{-N} = 1$, we find the N unique complex valued roots of 1 (unity) are

$$\gamma = 1^{\frac{1}{N}} = [e^{-2\pi i m}]^{\frac{1}{N}} \,, \tag{2.85}$$

for $m = 0, \ldots, N-1$. Taking $\gamma = \gamma^{(m)}$, the mth order root, the corresponding eigenvalues and eigenvectors are

$$\lambda^{(m)} = \sum_{t=0}^{N-1} c_t e^{-\frac{2\pi i m t}{N}} = \text{DFT}[c_t] = \tilde{c}_m \,, \tag{2.86}$$

$$\boldsymbol{u}^{(m)} = \frac{1}{\sqrt{N}}(1, e^{-\frac{2\pi i m}{N}}, \ldots, e^{-\frac{2\pi i m (N-1)}{N}})^\top \,. \tag{2.87}$$

Therefore the eigenvalues of C, represented in Eq. (2.86), are equal to the Discrete Fourier Transform (DFT) of the first column of C.[1]

We can thus express $C\boldsymbol{z}$ for any vector $\boldsymbol{z}$, using the DFT and its inverse, as follows:

$$C\boldsymbol{z} = \frac{1}{N} \sum_{t,m} e^{\frac{2\pi i m t}{N}} \tilde{c}_m z_t \tag{2.88}$$

$$= \frac{1}{N} \sum_{m} e^{\frac{2\pi i m t}{N}} \tilde{c}_m \tilde{z}_m = \text{DFT}^{-1}[\tilde{c}\tilde{z}] \,. \tag{2.89}$$

The product of C with a $2(N+1) \times 2(N+1)$ vector $\boldsymbol{z} = [\boldsymbol{w}, \boldsymbol{0}]^\top$, where $\boldsymbol{w}$ is any $N \times 1$ vector is

$$C\boldsymbol{z} = C \begin{bmatrix} \boldsymbol{w} \\ \boldsymbol{0} \end{bmatrix} = \begin{bmatrix} K\boldsymbol{w} \\ \boldsymbol{0} \end{bmatrix} \,. \tag{2.90}$$

[1]We write $\text{DFT}[c_t] = \tilde{c}_m$. The inverse discrete Fourier transform, $\text{DFT}^{-1}[\tilde{c}_m] = \frac{1}{N}\sum_{m=0}^{N-1} \tilde{c}_m e^{\frac{2\pi i t m}{N}} = c_t$.

The DFT and inverse DFT of an $M \times 1$ vector can be computed efficiently in $\mathcal{O}(M \log M)$ using fast Fourier transforms (FFTs). Thus $C\boldsymbol{z}$ can be used to exactly find $K\boldsymbol{w}$ for any $\boldsymbol{w}$ in $\mathcal{O}(N \log N)$ computations and $\mathcal{O}(N)$ memory!

Inverse matrix vector products, $K^{-1}\boldsymbol{w}$, can then be found in $\mathcal{O}(N \log N)$ computations, and $\mathcal{O}(N)$ storage using preconditioned conjugate gradients (PCG) (Atkinson, 2008), which exclusively make use of matrix vector products for solving linear systems. Although an exact solution is guaranteed after N iterations, typically a very small number of PCG iterations $J \ll N$ are required for convergence within machine precision. The required number of iterations for a practically exact solution is essentially independent of N, and depends more directly on the conditioning of the matrix K.

Thus exact Gaussian process inference can be achieved in $\mathcal{O}(N \log N)$ and $\mathcal{O}(N)$ memory, if K has Toeplitz structure, which is often the case for time series data. This compares to the standard $\mathcal{O}(N^3)$ computations and $\mathcal{O}(N^2)$ memory required using a Cholesky decomposition.

Inference, however, only solves half of the problem – hyperparameter learning, which requires many marginal likelihood evaluations – is often equally important. Toeplitz structure in K can be exploited to exactly evaluate $\log |K + \sigma^2 I|$ (Zohar, 1969); however, this operation requires $\mathcal{O}(N^2)$ computations and $\mathcal{O}(N)$ memory. A Matlab toolbox for Toeplitz operations can be found at `http://mloss.org/software/view/496/`. Turner (2010) and Cunningham et al. (2008) contain examples of Toeplitz methods applied to GPs.

Toeplitz methods can be applied to almost any of the models introduced in this thesis in the cases where stationary kernels are often used as components in a model (even if the overall kernel is non-stationary), and inputs are on a regular grid (e.g., time series, spatial statistics (with some modifications to the Toeplitz methods), etc.). For instance, these conditions apply to the econometrics and gene expression examples in chapter 3. We pursue Toeplitz methods further in section 4.4.

Gaussian Processes with Circulant Covariance Matrices, Discrete Bochner's Theorem, and the Empirical Spectral Density

One can show, using the identities presented in this section, that a circulant covariance matrix in the time (untransformed) domain becomes a diagonal matrix in the frequency (Fourier transformed) domain. For example,

$$C\boldsymbol{z} = \mathrm{DFT}^{-1}[\mathrm{diag}(\tilde{c})]\mathrm{DFT}[\boldsymbol{z}]\,, \tag{2.91}$$

as in Eq. (2.89). We will see in section 4.4.4 that the diagonalization of a circulant covariance matrix in the frequency space provides an efficient means of sampling from Gaussian processes with Toeplitz matrices.

Expressed differently,

$$c_\tau = \frac{1}{N}\sum_m e^{\frac{2\pi i m\tau}{N}}\tilde{c}_m\,. \tag{2.92}$$

Eq. (2.92) is in fact a discrete version of Bochner's theorem (section 2.4.1)! We could imagine modelling $\tilde{c}_m$ as a Gaussian process (with, for instance, a simple SE kernel), queried at the N points in the sum of Eq. (2.92), in order to learn a flexible circulant kernel, which could perform extrapolation, in the time domain. Such a model would transform an extrapolation problem in the time domain into an interpolation problem in the frequency domain.

The log marginal likelihood of a Gaussian process with a circulant covariance matrix (assuming now that the matrix has N rows), is

$$\log p(\boldsymbol{y}|X) = -\frac{1}{2}\boldsymbol{y}^\top C^{-1}\boldsymbol{y} - \frac{1}{2}\log|C| - \frac{N}{2}\log(2\pi)\,, \tag{2.93}$$

$$= -\frac{1}{2N}\sum_m \frac{|\tilde{y}_m|^2}{\tilde{c}_m} - \frac{1}{2}\sum_m \log(\tilde{c}_m) - \frac{N}{2}\log(2\pi)\,. \tag{2.94}$$

The circulant kernel in frequency space, $\boldsymbol{c}_m$, which maximizes the log likelihood in Eq. (2.94) is

$$\hat{c}(s_m) = \frac{|\tilde{y}(s_m)|^2}{N}\,, \tag{2.95}$$

and is known as the *empirical spectral density*, defined for frequencies $s_m = 0, f_s/N, 2f_s/N, \ldots, f_s/2$, where f_s is the sampling rate of the data. We first discussed the empirical spectral density in section 2.4.1. In chapter 4 we compare the spectral densities we learn using a spectral mixture (SM) kernel with the empirical spectral density of Eq. (2.95).

2.6.3.2 Kronecker Structure

Certain popular stationary kernels for multidimensional inputs, such as the squared exponential kernel, decompose as a product across input dimensions:

$$k(x_i, x_j) = \prod_{p=1}^{P} k^p(x_i^p, x_j^p)\,. \tag{2.96}$$

Assuming a product kernel as in Eq. (2.96), and inputs $x \in \mathcal{X}$ on a multidimensional grid $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_P \subset \mathbb{R}^P$,[1] the covariance matrix K decomposes into a Kronecker product of matrices over each input dimension $K = K^1 \otimes \cdots \otimes K^P$ (Saatchi, 2011).[2] The eigendecomposition of K into $QVQ^\top$ similarly decomposes: $Q = Q^1 \otimes \cdots \otimes Q^P$ and $V = V^1 \otimes \cdots \otimes V^P$. Each covariance matrix K^p in the Kronecker product has entries $K_{ij}^p = k^p(x_i^p, x_j^p)$ and decomposes as $K^p = Q^p V^p Q^{p\top}$. Thus the $N \times N$ covariance matrix K can be stored in $\mathcal{O}(PN^{\frac{2}{P}})$ and decomposed into $QVQ^\top$ in $\mathcal{O}(PN^{\frac{3}{P}})$ operations, for N datapoints and P input dimensions.[3] For a covariance matrix with Kronecker structure, this property thus greatly reduces the cost of both inference and learning.[4]

Kronecker methods were first used with Gaussian processes in Saatchi (2011). More detail on inference and learning using Kronecker structure, and details about how to relax the grid assumption, can be found in chapter 5, where we develop and exploit the structure in expressive kernels for large-scale multidimensional pattern extrapolation.

Indeed Kronecker methods can be directly applied to many of the models in this thesis, where we use product kernels on a grid.

[1]Note that this grid does not need to be evenly spaced.

[2]To be clear, we are indexing matrices using superscripts when describing Kronecker products $\otimes$. For example, K^j would be the j^{th} matrix in the Kronecker product, not the j^{th} power of K.

[3]The total number of datapoints $N = \prod_p |\mathcal{X}_p|$, where $|\mathcal{X}_p|$ is the cardinality of $\mathcal{X}_p$. For clarity of presentation, we assume each $|\mathcal{X}_p|$ has equal cardinality $N^{1/P}$.

[4]Note that the log determinant term in the marginal likelihood can be written as a sum of eigenvalues.

2.6.4 Multi-Output Gaussian Processes

It is unnatural to think of regression as finding a mapping from $f : \mathbb{R}^P \to \mathbb{R}^1$, $P \in \mathbb{N}$. There is no more reason to think, in general, that the output variable should be one dimensional, than there is to believe the function f only depends on one input variable. Real world problems often exhibit strong and sophisticated correlations between many output variables – particularly spatial coordinates in $\mathbb{R}^3$. For example, in positron emission tomography (PET) one wishes to localize the position of a particle emitting isotropic light – a regression from P photodetectors to a 3D position in space (Wilson, 2008). Thus we ought to generally be modelling functions $f : \mathbb{R}^P \to \mathbb{R}^W$, for any natural numbers P and W. The often held assumption that $W = 1$ is for convenience, and not because it truly reflects our beliefs about the world.

Indeed neural networks became popular partly because they allowed for sophisticated correlations between multiple outputs, through sharing adaptive hidden basis functions across the outputs. In taking the infinite limit of hidden units in a Bayesian neural network to derive a Gaussian process, as in section 2.4.5, these correlations vanish. In general, Gaussian process models do not naturally account for multiple correlated output variables.

Nonetheless, due to the prevalence of 'multiple output' (aka multi-task) regression problems – where it is entirely unsatisfactory to assume independence across outputs – there has been a rapidly growing effort to extend Gaussian process regression models to account for *fixed* correlations between output variables (Alvarez and Lawrence, 2011; Bonilla et al., 2008; Boyle and Frean, 2004; Teh et al., 2005; Yu et al., 2009).

In chapter 3 we introduce a scalable multi-output regression model, the Gaussian process regression network (GPRN), which accounts for *input varying* correlations between multiple outputs, to improve predictions about the values of each of these outputs.

2.6.5 Heteroscedastic Gaussian Processes

The predictive variance of a Gaussian process typically does not explicitly depend on the values of observations, only on their input locations, as shown in Eq. (2.27), except insomuch as kernel hyperparameters can be learned from data.[1] This is an undesirable property: if the variance of observations in a particular region of the input space is high, the predictive variance in that region should also be high.

Much like the assumption of uncorrelated outputs, the assumption of i.i.d. Gaussian noise is often grounded in convenience, rather than a true reflection of our beliefs. If we assume input dependent (aka *heteroscedastic*) noise, then our predictive variance at given test locations depends on the values of the nearby observations.

There has recently been an explosion of interest in extending Gaussian process regression to account for input dependent noise variances (Adams and Stegle, 2008; Goldberg et al., 1998; Kersting et al., 2007; Lázaro-Gredilla and Titsias, 2011; Turner, 2010; Wilson and Ghahramani, 2010a,b). In the case of multi-output regression, Wilson and Ghahramani (2010b, 2011) developed Gaussian process models to account for input dependent noise covariances (multivariate volatility) between multiple outputs. In chapter 3, we introduce the Gaussian process regression network (GPRN), which accounts for input dependent signal and noise correlations between multiple outputs, through an expressive input dependent combination of covariance kernels.

2.6.6 Sampling Kernel Hyperparameters

Kernel hyperparameters can be learned through marginal likelihood optimization, or integrated away using sampling methods (section 2.3.1.2). At the present time, marginal likelihood optimization is almost exclusively preferred, which at first glance is somewhat puzzling: since Gaussian processes (as kernel machines) are Bayesian methods, one would expect GP practitioners to be biased towards fully Bayesian treatments of kernel hyperparameters, which have been well known for

[1]Conversely, the predictive variance of a t-process *does* explicitly depend on the values of training observations (Shah et al., 2014).

decades (Rasmussen, 1996). Thus common practice suggests that a fully Bayesian treatment of hyperparameters may not be worthwhile.

Some potential reasons sampling kernel hyperparameters is not yet common practice include:

1. Historically any tuning of kernel hyperparameters – sampling or otherwise – has not been given the attention it deserves, given how significantly parameters like length-scale can affect predictions. This is perhaps partly because the more well known SVM framework only provides heuristics for kernel hyperparameter tuning, and heuristics are typically not emphasized in paper writing.

2. Sampling methods are not typically as computationally efficient, or as easy to implement, as optimization. Furthermore, it is not clear what sampling method ought to be implemented for kernel hyperparameters, although Hybrid Monte Carlo (HMC) and slice sampling are two common choices, as discussed in section 2.3.1.2.

3. Most importantly, the marginal likelihood surface is often sharply peaked for the kernels that are popular at the present time, such as the squared exponential kernel, which only use a small number of parameters (Rasmussen and Williams, 2006). In a range of experiments, Rasmussen (1996) found no significant empirical advantages to sampling over marginal likelihood optimisation for the squared exponential kernel.

However, there are a few niche areas, such as Bayesian optimization (Brochu et al., 2010), where sampling kernel hyperparameters has become standard. In such applications, predictive variances are particularly important for good performance, suggesting that sampling kernel hyperparameters can usefully improve predictive variances, even for standard kernels.

As discussed in section 2.6.2, we expect sophisticated kernels to become increasingly mainstream, in which case the marginal likelihood surface of Gaussian process models will generally become increasingly broad and multimodal, and sampling will become increasingly useful.

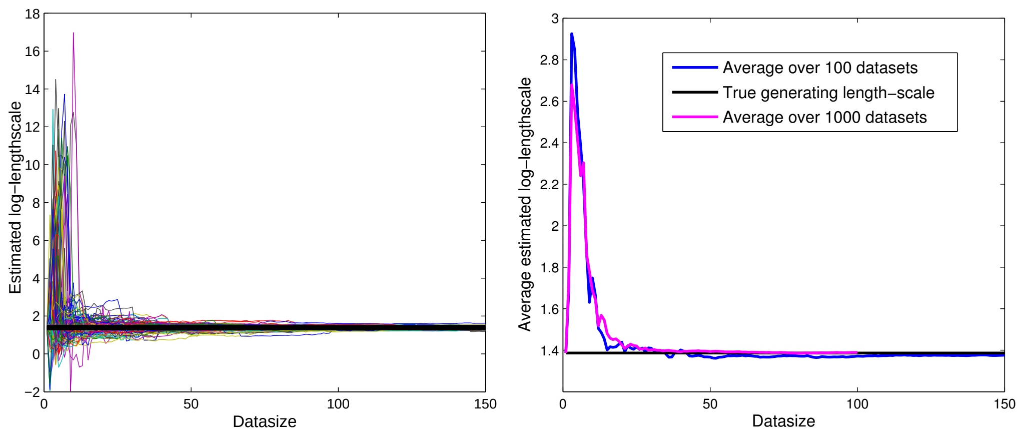
Overall, the biggest challenge in learning kernel hyperparameters in Gaussian process models, at the present time, arises when the Gaussian processes cannot be analytically marginalised. In this case one has to deal with the strong dependencies between Gaussian processes and their kernel hyperparameters. Perhaps the most successful resolution thus far is to approximate the marginal likelihood with an analytic expression, and optimize this approximate marginal likelihood with respect to hyperparameters (chapter 3).

Sampling kernel hyperparameters may also be useful when there are very few data points. GP marginal likelihood optimization is surprisingly biased towards length-scale overestimation, as over-fitting the covariance function corresponds to under-fitting in function space; and this effect is most pronounced on small datasets.

To exemplify this surprising under-fitting property of maximum marginal likelihood estimation of kernel hyperparameters, consider the following experiment. We sampled 100 datasets of size N = 150 from a GP with a squared exponential covariance function with a true length-scale of 4, signal standard deviation (a in Eq. (2.65)) of 1, and noise standard deviation 0.2, at 1 unit intervals. Using marginal likelihood optimization, I then estimated the kernel hyperparameters (length-scale, signal, and noise stdev) on each of these datasets, as a function of increasing datasize, initializing hyperparameters at their true values. Each curve in Figure 2.11a shows the estimated log length-scale, for a particular dataset, as a function of datasize. Figure 2.11b shows the learned length-scale, averaged in log-space, e.g. an estimate of $\mathbb{E}[\log \ell]$, not $\log \mathbb{E}[l]$, over the 100 datasets. The truth is in black. The trend is clearest in Figure 2.11b: there is a systematic length-scale overestimation (underfitting), which is mostly negligable after about $N = 20$ datapoints. In high dimensional input spaces, this under-fitting effect may be even more pronounced. As shown in Figure 2.11b, averaging 1000 datasets gives almost exactly the same results.

2.6.7 Non-Gaussian Processes

It is natural to wonder about nonparametric kernel machines which do not have any Gaussianity assumptions. Surely we could easily use other elliptical processes



(a) (b)

Figure 2.11: GP Underfitting via Marginal Likelihood Hyperparameter Estimation. a) Each curve represents the estimated length-scale of a Gaussian process for a particular dataset, at a given datasize. There are 100 datasets (and thus 100 different coloured curves). b) The log-lengthscale results in panel a) have been averaged to produce this figure. The results using 1000 datasets are shown in magenta, and they are similar to the results with 100 datasets. These figures consistently show length-scale overestimation, equivalently GP under-fitting, particularly for small $N < 20$ datasets. The standard deviation over the 1000 datasets follows the same trend as the magenta curve with a low of 0 and a high of 7.

parametrized by a covariance kernel – and after all, isn't the Gaussian distribution routinely criticized as too simplistic with unrealistically small tails?

However, the Gaussian assumption is not nearly as restrictive as it might initially seem. For example, Gaussian processes can essentially have support for *any* continuous function, and are universal approximators (Ghosal and Roy, 2006; van der Vaart and van Zanten, 2009). The choice of kernel is typically far more important than whether or not the process is truly Gaussian. Even when the Gaussian assumption clearly does not hold, it has not been a significant restriction in my experience.

Nonetheless, the Gaussian assumption obviously does not hold in many situations. For example, if data are known to be strictly positive, then a Gaussian distribution would erroneously assign negative values non-zero probability. In such cases, often a log transform is applied to the data as a preprocessing step, and then a Gaussian process is used on the transformed data. However, this procedure assumes the data are lognormal distributed, which will likely also be incorrect. Snelson et al. (2003) propose to marginally transform a Gaussian process, and to learn the transformation as part of Bayesian inference; this procedure is equivalent to pre-processing the data with some unknown function and then modelling the transformed data with a Gaussian process, having learned the transformation from data.

However, the joint dependency structure of the function may also be non-Gaussian. For example, a process may have *tail-dependence*, where extreme values of $f(x_p)$ are highly correlated with extreme values of $f(x_q)$ for inputs $x_p, x_q \in \mathcal{X}$. Wilson and Ghahramani (2010a) introduce the *copula process*, which separates the marginal distributions of a stochastic process from its joint dependency structure.

It is intuitive that much of the standard Gaussian process machinery could be straightforwardly extended to more general elliptical processes – where any collection of function values has a joint elliptical distribution with a covariance matrix parametrized by a kernel. At present, however, the theoretical and practical benefits of elliptical processes, such as t-processes, are not well understood; for example, Rasmussen and Williams (2006) wonder whether "the t-process is perhaps not as exciting as one might have hoped".

In response to these questions, the recent paper Shah et al. (2014) deeply explores t-processes and elliptical processes in general, and shows that t-processes can be meaningfully more flexible than Gaussian processes, without additional computational limitations. In particular, the predictive variances of a t-process explicitly depend on the values of training observations – an intuitively desirable property – and are generally of higher quality than the GP predictive variances. This property of the t-process is especially useful in growing GP applications such as Bayesian optimisation (Brochu et al., 2010), where high quality predictive variances are particularly important for efficiently finding global optima.

In the future it is reasonable to expect departures away from Gaussianity to become more common, since there can be very little computational expense in relaxing the Gaussian assumption, and we will become increasingly confident about what (non-Gaussian) distribution more truly describes the data in many specialized applications.

Chapter 3

Gaussian Process Regression Networks

In this chapter we introduce a new regression framework, Gaussian Process Regression Networks (GPRNs), which unifies many of the themes identified in chapter 2 as promising areas of future Gaussian process research: multi-output regression, multivariate heteroscedasticity, non-parametric modelling, scalable inference, non-Gaussian predictive distributions, pattern discovery, and flexible kernels. Underlying all of the properties of the GPRN is a highly expressive covariance kernel, which combines the structural properties of Bayesian neural networks with the non-parametric flexibility of Gaussian processes.

Each output variable in the Gaussian process regression network is an input dependent mixture of latent Gaussian process basis functions, where the mixing weights are themselves Gaussian processes. Thus, conditioned on the mixing weights, each output is a Gaussian process with a kernel that is an input dependent mixture of the kernels used for each basis function. Even if the kernels used in each basis function are simple, a shared adaptive mixture of kernels can be highly expressive – allowing each individual output (response) variable to capture enlightening shifts in covariance structure, and enabling expressive input dependent correlations across each output. As we will see, these expressive kernels can be used to discover meaningful features in our data, such a geological structures which can affect how correlations between different metals vary with geographical location.

Moreover, the structure of the GPRN can be exploited for 1) input dependent noise correlations between outputs (multivariate heteroscedasticity), at no additional computational cost, and 2) inference that scales *linearly* with the total number of output (response) variables, as opposed to the cubic scaling that is common for multi-output Gaussian process models (which, incidentally, do not typically model *input dependent* correlations).

The GPRN is an example of how one can construct expressive kernels for fast automatic pattern discovery with Gaussian processes. Moreover, the idea of an adaptive mixture of latent basis functions extends beyond Gaussian processes – this general architecture, which we henceforth refer to as an *adaptive network*, is a powerful machine learning tool even if the basis functions and mixing functions are not Gaussian processes. Indeed, adaptive networks could usefully be viewed as an extension of standard neural network architectures, and like neural networks, could have enlightening analogues in biological learning. We further discuss adaptive networks, and some potential applications, in section 3.9.

This chapter extends the material in my papers Wilson et al. (2011, 2012), which first introduced the GPRN. The material on variational Bayes inference was largely contributed by David A. Knowles. Since the introduction of the GPRN, Nguyen and Bonilla (2013) have derived particularly efficient variational inference especially designed for Gaussian process regression networks. The proposed NMR application extends the model of Wilson et al. (2014) to account for time-varying reactions.

In section 3.8 we discuss the connection between the GPRN and the generalised Wishart process (GWP) (Wilson and Ghahramani, 2010b, 2011), a nonparametric process over matrices which evolves over time, space, or another covariate. Wilson and Ghahramani (2012) discuss both GWPs and GPRNs in the context of modelling input dependent correlations between multiple output variables.

The benefits of the GPRN framework introduced in this chapter, and the closed form spectral mixture (SM) kernels of chapter 4, are largely complementary. Indeed, SM kernels can be used as the basis function kernels within the GPRN framework, to form powerful new non-stationary kernels, which we discuss further in section 4.4 of the next chapter.

3.1 Introduction

Neural networks became popular largely because they shared adaptive hidden basis functions across multiple outputs, which allowed for correlations between the outputs, and the ability to discover hidden features in data. By contrast, Gaussian processes with standard kernels, though effective at many regression and classification problems, are simply smoothing devices (MacKay, 1998), and do not naturally handle correlations between multiple output variables.

Recently there has been an explosion of interest in extending the Gaussian process regression framework to account for *fixed* correlations between output variables (Alvarez and Lawrence, 2011; Bonilla et al., 2008; Boyle and Frean, 2004; Teh et al., 2005; Yu et al., 2009). These are often called 'multi-task' learning or 'multiple output' regression models. Capturing correlations between outputs (responses) can be used to make better predictions. Imagine we wish to predict cadmium concentrations in a region of the Swiss Jura, where geologists are interested in heavy metal concentrations. A standard Gaussian process regression model would only be able to use cadmium training measurements. With a multi-task method, we can also make use of correlated heavy metal measurements to enhance cadmium predictions (Goovaerts, 1997). We could further enhance predictions if we could use how these (signal) correlations change with geographical location.

There has similarly been great interest in extending Gaussian process (GP) regression to account for input dependent noise variances (Adams and Stegle, 2008; Goldberg et al., 1998; Kersting et al., 2007; Lázaro-Gredilla and Titsias, 2011; Turner, 2010; Wilson and Ghahramani, 2010a,b). Wilson and Ghahramani (2010b, 2011) further extended the GP framework to accommodate input dependent noise correlations between multiple output (response) variables, with the *generalised Wishart process* (section 3.8).

In this chapter, we introduce a new regression framework, Gaussian Process Regression Networks (GPRN), which combines the structural properties of Bayesian neural networks with the nonparametric flexibility of Gaussian processes. This network is an adaptive mixture of Gaussian processes, which naturally accommodates input dependent signal and noise correlations between multiple output variables, input dependent length-scales and amplitudes, and heavy tailed predictive distributions, without expensive or numerically unstable computations. The GPRN framework extends and unifies the work of Journel and Huijbregts (1978), Neal (1996), Gelfand et al. (2004), Teh et al. (2005), Adams and Stegle (2008), Turner (2010), and Wilson and Ghahramani (2010a, 2011).

Throughout this chapter we assume we are given a dataset of input output pairs, $\mathcal{D} = \{(x_i, \boldsymbol{y}(x_i)) : i = 1, \dots, N\}$, where $x \in \mathcal{X}$ is an input (predictor) variable belonging to an arbitrary set $\mathcal{X}$, and $\boldsymbol{y}(x)$ is the corresponding p dimensional output; each element of $\boldsymbol{y}(x)$ is a one dimensional output (response) variable, for example the concentration of a single heavy metal at a geographical location x. We aim to predict $\boldsymbol{y}(x_*)|x_*, \mathcal{D}$ and $\Sigma(x_*) = \text{cov}[\boldsymbol{y}(x_*)|x_*, \mathcal{D}]$ at a test input x_*, while accounting for input dependent signal and noise correlations between the elements of $\boldsymbol{y}(x)$.

We start by introducing the GPRN framework and discussing inference and computational complexity, in sections 3.2, 3.3, and 3.4, respectively. We describe related work in depth (section 3.5), before comparing to eight multiple output GP models, on geostatistics and large-scale gene expression datasets, and three multivariate volatility models on several benchmark financial datasets.[1] We then discuss connections with the generalised Wishart process (Wilson and Ghahramani, 2010b, 2011) in section 3.8, and a generalisation of the Gaussian process regression network to adaptive networks, with example applications in nuclear magnetic resonance (NMR) spectroscopy, ensemble learning, and changepoint modelling, in section 3.9. The NMR spectroscopy application is related to work in preparation (Wilson et al., 2014).

[1]appendix A on time series introduces much of the background material needed to fully appreciate the financial and multivariate volatility experiments.

3.2 Model Specification

We wish to model a p dimensional function $\boldsymbol{y}(x)$, with signal and noise correlations that vary with x in an arbitrary input space $\mathcal{X}$ (although we are typically interested in $x \in \mathbb{R}^M$, for $M \in \mathbb{Z}^+$).

We model $\boldsymbol{y}(x)$ as

$$\boldsymbol{y}(x) = W(x)[\boldsymbol{f}(x) + \sigma_f \boldsymbol{\epsilon}] + \sigma_y \boldsymbol{z}\,, \tag{3.1}$$

where $\boldsymbol{\epsilon} = \boldsymbol{\epsilon}(x)$ and $\boldsymbol{z} = \boldsymbol{z}(x)$ are respectively $\mathcal{N}(0, I_q)$ and $\mathcal{N}(0, I_p)$ white noise processes.[1] I_q and I_p are $q \times q$ and $p \times p$ dimensional identity matrices. $W(x)$ is a $p \times q$ matrix of independent Gaussian processes such that $W(x)_{ij} \sim \mathcal{GP}(0, k_w)$, and $\boldsymbol{f}(x) = (f_1(x), \dots, f_q(x))^\top$ is a $q \times 1$ vector of independent GPs with $f_i(x) \sim \mathcal{GP}(0, k_{f_i})$. The GPRN prior on $\boldsymbol{y}(x)$ is induced through GP priors in $W(x)$ and $\boldsymbol{f}(x)$, and the noise model is induced through $\boldsymbol{\epsilon}$ and $\boldsymbol{z}$.

We represent the *Gaussian process regression network* (GPRN)[2] of Equation (3.1) in Figure 3.1. Each of the latent Gaussian processes in $\boldsymbol{f}(x)$ has additive Gaussian noise. Changing variables to include the noise $\sigma_f \boldsymbol{\epsilon}$, we let $\hat{f}_i(x) = f_i(x) + \sigma_f \epsilon \sim \mathcal{GP}(0, k_{\hat{f}_i})$, where

$$k_{\hat{f}_i}(x_a, x_w) = k_{f_i}(x_a, x_w) + \sigma_f^2 \delta_{aw}\,, \tag{3.2}$$

and δ_{aw} is the Kronecker delta. The latent *node functions* $\hat{\boldsymbol{f}}(x)$ are connected together to form the outputs $\boldsymbol{y}(x)$. The strengths of the connections change as a function of x; the weights themselves – the entries of $W(x)$ – are functions.[3] Old connections can break and new connections can form. This is an *adaptive* network, where the signal and noise correlations between the components of $\boldsymbol{y}(x)$ vary with x. We label the length-scale hyperparameters for the kernels k_w and k_{f_i} as $\boldsymbol{\theta}_w$ and $\boldsymbol{\theta}_f$ respectively. We often assume that all the weight GPs share

[1]We have not explicitly written the functional dependence of $\boldsymbol{\epsilon}$ and $\boldsymbol{z}$ on x in Eq. (3.1) to emphasize that these 'noise parameters' do not have an input dependent variance.

[2]Coincidentally, there is an unrelated paper called "Gaussian process networks" (Friedman and Nachman, 2000), which is about learning the structure of Bayesian networks – e.g. the direction of dependence between random variables.

[3]When $W(x)$ is a matrix of constants $W(x) \to W$, we essentially recover the semiparametric latent factor (SLFM) model of Teh et al. (2005). A detailed discussion of related models is in section 3.5.

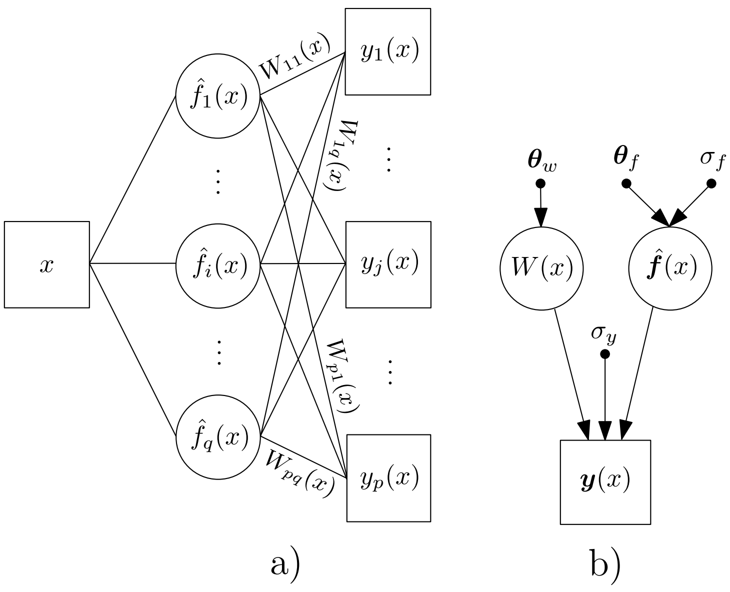


Figure 3.1: Structure of the Gaussian process regression network. Latent random variables and observables are respectively labelled with circles and squares, except for the weight functions in a). a) This neural network style diagram shows the q components of the vector $\hat{\boldsymbol{f}}$ (GPs with additive noise), and the p components of the vector $\boldsymbol{y}$. The links in the graph, four of which are labelled, are latent random weight *functions*. Every quantity in this graph depends on the input x. This graph emphasises the adaptive nature of this network: links can change strength or even disappear as x changes. b) A directed graphical model showing the generative procedure with relevant variables. Hyperparameters are labelled with dots.

the same covariance kernel k_w, including hyperparameters. Roughly speaking, sharing length-scale hyperparameters amongst the weights means that, a priori, the strengths of the connections in Figure 3.1 vary with x at roughly the same rate.

To explicitly separate the adaptive signal and noise correlations, we re-write (3.1) as

$$\boldsymbol{y}(x) = \underbrace{W(x)\boldsymbol{f}(x)}_{\text{signal}} + \underbrace{\sigma_f W(x)\boldsymbol{\epsilon} + \sigma_y \boldsymbol{z}}_{\text{noise}} \, . \tag{3.3}$$

Given $W(x)$, each of the outputs $\boldsymbol{y}_i(x)$, $i = 1, \dots, p$, is a Gaussian process with kernel

$$k_{y_i}(x_a, x_w) = \sum_{j=1}^{q} W_{ij}(x_a) k_{\hat{f}_j}(x_a, x_w) W_{ij}(x_w) + \delta_{aw}\sigma_y^2 \, , \tag{3.4}$$

which follows from Eqs. (3.2) and (3.1).

The components of $\boldsymbol{y}(x)$ are coupled through the matrix $W(x)$. Training the network involves conditioning $W(x)$ on the data $\mathcal{D}$, and so the predictive covariances of $\boldsymbol{y}(x_*)|\mathcal{D}$ are now influenced by the values of the observations, and not just distances between the test point x_* and the observed points $x_1, \dots, x_N$, as is the case for independent GPs.

We can view (3.4) as an adaptive kernel learned from the data. There are several other interesting features in equation (3.4): 1) the amplitude of the covariance function, $\sum_{j=1}^{q} W_{ij}(x)W_{ij}(x')$, is non-stationary (input dependent); 2) even if each of the kernels k_{f_j} has different *stationary* length-scales, the mixture of the kernels k_{f_j} is input dependent and so the effective overall length-scale is non-stationary; 3) the kernels k_{f_j} may be entirely different: some may be periodic, others squared exponential, others Brownian motion, and so on. Therefore the overall covariance kernel may be continuously switching between regions of entirely different covariance structures.

In addition to modelling signal correlations, we can see from Equation (3.3) that the GPRN is also a multivariate volatility model. The noise covariance is $\sigma_f^2 W(x)W(x)^\top + \sigma_y^2 I_p$. Since the entries of $W(x)$ are GPs, this noise model is an example of a *generalised Wishart process* (Wilson and Ghahramani, 2010b, 2011) (discussed further in section 3.8).

The number of nodes q influences how the model accounts for signal and noise correlations. If q is smaller than p, the dimension of $\boldsymbol{y}(x)$, the model performs dimensionality reduction and matrix factorization as part of the regression on $\boldsymbol{y}(x)$ and $\mathrm{cov}[\boldsymbol{y}(x)]$. However, we may want $q > p$, for instance if the output space were one dimensional ($p = 1$). In this case we would need $q > 1$ for nonstationary length-scales and covariance structures. For a given dataset, we can vary q and select the value which gives the highest marginal likelihood on training data.

Note that the signal and noise components of the model are encouraged to increase and decrease together with the magnitude of the weight matrix $W(x)$. It turns out this is a crucial inductive bias for extracting meaningful structure with the matrix $W(x)$ in models like the GPRN. We have found, for instance, that if the signal is modelled by $W(x)f(x)$, but the noise is modelled by a separate matrix $G(x)$ (and not influenced by $W(x)$, then it is difficult to extract useful information from the data – because such a model is highly flexible but lacks reasonable assumptions.

3.3 Inference

We have specified a prior $p(\boldsymbol{y}(x))$ at all points x in the domain $\mathcal{X}$, and a noise model, so we can infer the posterior $p(\boldsymbol{y}(x)|\mathcal{D})$. The prior on $\boldsymbol{y}(x)$ is induced through the GP priors in $W(x)$ and $\boldsymbol{f}(x)$, and the parameters $\boldsymbol{\gamma} = \{\boldsymbol{\theta}_f, \boldsymbol{\theta}_w, \sigma_f, \sigma_y\}$. We perform inference directly over the GPs and parameters.

We explicitly re-write the prior over GPs in terms of $\boldsymbol{u} = (\hat{\mathbf{f}}, \mathbf{W})$, a vector composed of all the node and weight Gaussian process functions, evaluated at the training points $\{x_1, \dots, x_N\}$. There are q node functions and $p \times q$ weight functions. Therefore

$$p(\boldsymbol{u}|\sigma_f, \boldsymbol{\theta}_f, \boldsymbol{\theta}_w) = \mathcal{N}(0, C_B)\,, \tag{3.5}$$

where C_B is an $Nq(p+1) \times Nq(p+1)$ block diagonal matrix, since the weight and node functions are a priori independent. We order the entries of $\boldsymbol{u}$ so that the first q blocks are $N \times N$ covariance matrices $K_{\hat{f}_i}$ from the node kernels $k_{\hat{f}_i}$, and the last blocks are $N \times N$ covariance matrices K_w from the weight kernel k_w.

From (3.1), the likelihood is

$$p(\mathcal{D}|\boldsymbol{u}, \sigma_y) = \prod_{i=1}^{N} \mathcal{N}(\boldsymbol{y}(x_i); W(x_i)\hat{\boldsymbol{f}}(x_i), \sigma_y^2 I_p)\,. \tag{3.6}$$

Applying Bayes' theorem,

$$p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma}) \propto p(\mathcal{D}|\boldsymbol{u}, \sigma_y)p(\boldsymbol{u}|\sigma_f, \boldsymbol{\theta}_f, \boldsymbol{\theta}_{\boldsymbol{w}})\,. \tag{3.7}$$

We sample from the posterior in (3.7) using elliptical slice sampling (ESS) (Murray et al., 2010), which is specifically designed to sample from posteriors with strongly correlated Gaussian priors. For comparison we approximate (3.7) using a message passing implementation of variational Bayes (VB). We also use VB to learn the hyperparameters $\boldsymbol{\gamma}|\mathcal{D}$. In the next sections we describe the ESS and VB inference in greater detail.

By incorporating noise on $\boldsymbol{f}$, the GP network accounts for input dependent noise correlations (as in (3.3)), without the need for costly or numerically unstable matrix decompositions during inference. The matrix $\sigma_y^2 I_p$ does not change with x and requires only one $\mathcal{O}(1)$ operation to invert. In a more typical multivariate volatility model, one must decompose a $p \times p$ matrix $\Sigma(x)$ once for each datapoint x_i (N times in total), an $\mathcal{O}(Np^3)$ operation which is prone to numerical instability. In general, multivariate volatility models are intractable for $p > 5$ (Engle, 2002; Gouriéroux et al., 2009). Moreover, multi-task Gaussian process models typically have an $\mathcal{O}(N^3p^3)$ complexity (Alvarez and Lawrence, 2011). In section 3.4 we show that, fixing the number of ESS or VB iterations, GPRN inference scales linearly with the number of input dimensions p. This scaling is possible because of the inference in this section, which exploits the structure of the GPRN. Similar scaling could be achieved with the popular SLFM (Teh et al., 2005) and LMC (Journel and Huijbregts, 1978) multi-task models through exploiting model structure in inference in a similar way.

We note that it is possible to reduce the number of modes in the posterior over the weights W and nodes $\mathbf{f}$ by constraining W or $\mathbf{f}$ to be positive. For MCMC it is straightforward to do this by exponentiating the weights, as in Adams and Stegle (2008), Turner (2010) and Adams et al. (2010). For VB it is more straightforward to explicitly constrain the weights to be positive using a truncated Gaussian representation. We found that these extensions did not significantly improve empirical performance, although exponentiating the weights sometimes improved numerical stability for MCMC on the multivariate volatility experiments. For Adams and Stegle (2008) exponentiating the weights will have been more valuable because they use Expectation Propagation, which in their case would centre probability mass between symmetric modes. MCMC and VB approaches are more robust to this problem. MCMC can explore these symmetric modes, and VB will concentrate on one of these modes without losing the expressivity of the GPRN prior.

3.3.1 Elliptical Slice Sampling

To sample from $p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma})$, we could use a Gibbs sampling scheme which would have conjugate posterior updates, alternately conditioning on weight and node functions. However, this Gibbs cycle would mix poorly because of the correlations between the weight and node functions in the posterior $p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma})$. In general, MCMC samples from $p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma})$ mix poorly because of the strong correlations in the prior $p(\boldsymbol{u}|\sigma_f, \boldsymbol{\theta}_f, \boldsymbol{\theta}_w)$ imposed by C_B. The sampling process is also often slowed by costly matrix inversions in the likelihood.

We use Elliptical Slice Sampling (Murray et al., 2010), a recent MCMC technique specifically designed to sample from posteriors with tightly correlated Gaussian priors. ESS performs joint updates and has no free parameters. Since there are no costly or numerically unstable matrix inversions in the likelihood of Eq. (3.6) we also find sampling to be efficient.

With a sample from $p(\boldsymbol{u}|\mathcal{D}, \boldsymbol{\gamma})$, we can sample from the predictive $p(W(x_*), \boldsymbol{f}(x_*)|\boldsymbol{u}, \sigma_f, \mathcal{D})$. Let $W_*^i, \boldsymbol{f}_*^i$ be the ith such joint sample. Using our generative GPRN model we can then construct samples of $p(\boldsymbol{y}(x_*)|W_*^i, \boldsymbol{f}_*^i, \sigma_f, \sigma_y)$, from which we can construct the predictive distribution

$$p(\boldsymbol{y}(x_*)|\mathcal{D}) = \lim_{J\to\infty} \frac{1}{J} \sum_{i=1}^{J} p(\boldsymbol{y}(x_*)|W_*^i, \boldsymbol{f}_*^i, \sigma_f, \sigma_y)\,. \tag{3.8}$$

We see that even with a Gaussian observation model, the predictive distribution in (3.8) is an infinite mixture of Gaussians, and will generally be heavy tailed. Since samples of $W_*^i, \boldsymbol{f}_*^i$ can take any value, the joint posterior $p(W(x_*), \boldsymbol{f}(x_*)|\boldsymbol{u}, \sigma_f, \mathcal{D})$ is highly non-Gaussian, and mixtures of Gaussians are dense in the set of probability distributions, the GPRN has the potential to be highly flexible.

Mixing was assessed by looking at trace plots of samples, and the likelihoods of these samples. Specific information about how long it takes to sample a solution for a given problem is in the experiments of section 3.6.

3.3.2 Variational Inference for the GPRN

We perform variational EM (Jordan et al., 1999) to fit an approximate posterior q to the true posterior p, by minimising the Kullback-Leibler divergence $KL(q||p) = -H[q(\mathbf{v})] - \int q(\mathbf{v}) \log p(\mathbf{v}) d\mathbf{v}$, where $H[q(\mathbf{v})] = -\int q(\mathbf{v}) \log q(\mathbf{v}) d\mathbf{v}$ is the entropy and $\mathbf{v} = \{\mathbf{f}, \mathbf{W}, \sigma_f^2, \sigma_y^2, a_j\}$.

E-step. We use Variational Message Passing (Ghahramani and Beal, 2001; Winn and Bishop, 2006) under the Infer.NET framework (Minka et al., 2010) to estimate the posterior over $\mathbf{v} = \{\mathbf{f}, \mathbf{W}, \sigma_f^2, \sigma_y^2, a_j\}$, where the $\{a_j\}$ are signal variance hyperparameters for each node function j, so that $k_{\hat{f}_j} \to a_j k_{\hat{f}_j}$.

We specify inverse Gamma priors on $\{\sigma_f^2, \sigma_y^2, a_j\}$:

$$\sigma_{fj}^2 \sim \mathrm{IG}(\alpha_{\sigma_f^2}, \beta_{\sigma_f^2}),\ \sigma_y^2 \sim \mathrm{IG}(\alpha_{\sigma_y^2}, \beta_{\sigma_y^2}),\ a_j \sim \mathrm{IG}(\alpha_a, \beta_a).$$

We use a variational posterior of the following form:

$$q(\mathbf{v}) = q_{\sigma_y^2}(\sigma_y^2) \prod_{j=1}^{Q} q_{\mathbf{f}_j}(\mathbf{f}_j) q_{\sigma_{fj}^2}(\sigma_{fj}^2) q_{a_j}(a_j) \prod_{i=1}^{P} q_{\mathbf{W}_{ij}}(\mathbf{W}_{ij}) \prod_{n=1}^{N} q_{\hat{f}_{nj}}(\hat{f}_{nj})$$

where $q_{\sigma_y^2}, q_{\sigma_{fj}^2}$ and q_{a_j} are inverse Gamma distributions; $q_{\hat{f}_{nj}}$ is a univariate normal distribution (indexed by training datapoint number n); and $q_{\mathbf{f}_j}$ and $q_{\mathbf{W}_{ij}}$ are N dimensional multivariate normal distributions (operating over the N training points). We have included the noise free variables $\mathbf{f}$ and the noisy variables $\hat{f}$. The lengthscales $\{\theta_f, \theta_w\}$ do not appear here because they are optimised in the M-step below (we could equivalently consider a point mass "distribution" on the lengthscales).

For mathematical and computational convenience we introduce the following variables which are deterministic functions of the existing variables in the model:

$$w_{nij} := W_{ij}(x_n), \qquad f'_{nj} := f_j(x_n) \tag{3.9}$$

$$t_{nij} := w_{nij} \hat{f}_{nj}, \qquad s_{in} := \sum_j t_{nij} \tag{3.10}$$

We refer to these as "derived" variables. Note that the observations $y_i(x_n) \sim \mathcal{N}(s_{in}, \sigma_y^2)$ and that $\hat{f}_{nj} \sim \mathcal{N}(f'_{nj}, \sigma_{f_j}^2)$. These derived variables are all given univariate normal "pseudo-marginals" which Variational message passing uses as conduits to pass appropriate moments, resulting in the same updates as standard VB (see Winn and Bishop (2006) for details). Using the derived variables the full model can be written as

$$\begin{aligned} p(\mathbf{v}) \propto \mathrm{IG}(\sigma_y^2; \alpha_{\sigma_y^2}, \beta_{\sigma_y^2}) \prod_{j=1}^{Q} \Bigg(\mathcal{N}(\mathbf{f}_j; 0, a_j K_{f_j}) \\ \mathrm{IG}(\sigma_{fj}^2; \alpha_{\sigma_f^2}, \beta_{\sigma_f^2}) \mathrm{IG}(a_j; \alpha_a, \beta_a) \prod_{i=1}^{P} \Bigg[\mathcal{N}(\mathbf{W}_{ij}; 0, K_w) \\ \prod_{n=1}^{N} \delta(w_{nij} - W_{ij}(x_n)) \delta(f'_{nj} - \hat{f}_j(x_n)) \mathcal{N}(\hat{f}_{nj}; f'_{nj}, \sigma_{f_j}^2) \\ \delta(t_{nij} - w_{nij} \hat{f}_{nj}) \delta(s_{in} - \sum_j t_{nij}) \mathcal{N}(y_i(x_n); s_{in}, \sigma_y^2) \Bigg] \Bigg) \end{aligned}$$

The updates for $\mathbf{f}, \mathbf{W}, \sigma_f^2, \sigma_y^2$ are standard VB updates and are available in Infer.NET. The update for the ARD parameters a_j however required specific implementation. The factor itself is

$$\begin{aligned} \log \mathcal{N}(\mathbf{f}_j; \mathbf{0}, a_j K_f) &\stackrel{c}{=} -\frac{1}{2} \log |a_j K_j| - \frac{1}{2} \mathbf{f}_j^T (a_j K_f)^{-1} \mathbf{f}_j \\ &= -\frac{N}{2} \log a_j - \frac{1}{2} \log |K_j| - \frac{1}{2} a_j^{-1} \mathbf{f}_j^T K_f^{-1} \mathbf{f}_j \end{aligned} \tag{3.11}$$

where $\stackrel{c}{=}$ denotes equality up to an additive constant. Taking expectations with respect to $\mathbf{f}$ under q we obtain the VMP message to a_j as IG $\left(a_j; \frac{N}{2} - 1, \frac{1}{2} \langle \mathbf{f}_j^T K_f^{-1} \mathbf{f}_j \rangle\right)$. Since the variational posterior on $\mathbf{f}$ is multivariate normal the expectation $\langle \mathbf{f}_j^T K_f^{-1} \mathbf{f}_j \rangle$ is straightforward to calculate.

M-step. In the M-step we optimise the variational lower bound with respect to the log length scale parameters $\{\theta_f, \theta_w\}$, using gradient descent with line search. When optimising θ_f we only need to consider the contribution to the lower bound of the factor $\mathcal{N}(\mathbf{f}_j; 0, a_j K_{f_j})$ (see (3.11)), which is straightforward to evaluate and differentiate. From (3.11) we have:

$$\begin{aligned}&\langle \log \mathcal{N}(\mathbf{f}_j; 0, a_j K_{f_j}) \rangle_q \\ &\overset{c}{=} -\frac{N}{2} \log a_j - \frac{1}{2} \log |K_{f_j}| - \frac{1}{2} \langle a_j^{-1} \rangle \langle \mathbf{f}_j^T K_{f_j}^{-1} \mathbf{f}_j \rangle\end{aligned}$$

We will need the gradient with respect to θ_f:

$$\begin{aligned}&\frac{\partial \langle \log \mathcal{N}(\mathbf{f}_j; 0, a_j K_{f_j}) \rangle}{\partial \theta_f} \\ &= -\frac{1}{2} \mathrm{tr} \left(K_{f_j}^{-1} \frac{\partial K_{f_j}}{\partial \theta_f} \right) - \frac{1}{2} \langle a_j^{-1} \rangle \langle \mathbf{f}_j^T K_{f_j}^{-1} \frac{\partial K_{f_j}}{\partial \theta_f} K_{f_j}^{-1} \mathbf{f}_j \rangle\end{aligned}$$

The expectations here are straightforward to compute analytically since $\mathbf{f}_j$ has multivariate normal variational posterior. Analogously for θ_w we consider the contribution of $\mathcal{N}(\mathbf{W}_{pq}; 0, K_W)$.

VB predictive distribution. The predictive distribution for the output $\mathbf{y}^*(x)$ at a new input location x is calculated as

$$p(\mathbf{y}^*(x)|\mathcal{D}) = \int p(\mathbf{y}^*(x)|W(x), f(x)) p(W(x), f(x)|\mathcal{D}) dW df \tag{3.12}$$

VB fits the approximation $p(W(x), f(x)|\mathcal{D}) = q(W)q(f)$, so the approximate predictive is

$$p(\mathbf{y}^*(x)|\mathcal{D}) = \int p(\mathbf{y}^*(x)|W(x), \hat{f}(x)) q(W) q(\hat{f}) dW d\hat{f} \tag{3.13}$$

We can calculate the mean and covariance of this distribution analytically:

$$\bar{\mathbf{y}}^*(x)_i = \sum_k \mathbb{E}(W_{ik}^*) \mathbb{E}[\hat{f}_k^*] \tag{3.14}$$

$$\mathrm{cov}(\mathbf{y}^*(x))_{ij} = \sum_k [\mathbb{E}(W_{ik}^*) \mathbb{E}(W_{jk}^*) \mathrm{var}(\hat{f}_k^*) + \delta_{ij} \mathrm{var}(W_{ik}^*) \mathbb{E}(\hat{f}_k^{*2})] + \delta_{ij} \mathbb{E}[\sigma_y^2] \tag{3.15}$$

where $\delta_{ij} = \mathbb{I}[i = j]$ is the Kronecker delta function, $W_{ik}^* = W_{ik}(x)$ and $\hat{f}_k^* = \hat{f}_k(x)$. The moments of W_{ik}^* and $\hat{f}_k^*$ under q are straightforward to obtain from $q(W)$ and $q(f)$ respectively using the standard GP prediction equations (chapter 2). It is also of interest to calculate the *noise* covariance. Recall our model can be written as

$$\boldsymbol{y}(x) = \underbrace{W(x)\boldsymbol{f}(x)}_{\text{signal}} + \underbrace{\sigma_f W(x)\boldsymbol{\epsilon} + \sigma_y \boldsymbol{z}}_{\text{noise}} \tag{3.16}$$

Let $\boldsymbol{n}(x) = \sigma_f W(x)\boldsymbol{\epsilon} + \sigma_y \boldsymbol{z}$ be the noise component. The covariance of $\boldsymbol{n}(x)$ under q is then

$$\text{cov}(\boldsymbol{n}(x))_{ij} = \sum_k \mathbb{E}[\sigma_{f_k}^2][\mathbb{E}(W_{ik}^*)\mathbb{E}(W_{jk}^*) + \delta_{ij}\text{var}(W_{jk}^*)] + \delta_{ij}\mathbb{E}[\sigma_y^2] \tag{3.17}$$

3.4 Computational Considerations

The computational complexity of a Markov chain Monte Carlo GPRN approach is mainly limited by taking the Cholesky decomposition of the block diagonal C_B, an $Nq(p+1) \times Nq(p+1)$ covariance matrix in the prior on GP function values ($\boldsymbol{u} \sim \mathcal{N}(0, C_B)$). But pq of these blocks are the same $N \times N$ covariance matrix K_w for the weight functions, and q of these blocks are the covariance matrices $K_{\hat{f}_i}$ associated with the node functions, and $\text{chol}(\text{blkdiag}(A, B, \dots)) = \text{blkdiag}(\text{chol}(A), \text{chol}(B), \dots)$. Therefore assuming the node functions share the same covariance function (which they do in our experiments), the complexity of this operation is only $\mathcal{O}(N^3)$, the same as for regular Gaussian process regression. At worst this complexity is $\mathcal{O}(qN^3)$, assuming different covariance functions for each node.

Sampling also requires likelihood evaluations. Since there are input dependent noise correlations between the elements of the p dimensional observations $\boldsymbol{y}(x_i)$, multivariate volatility models would normally require inverting[1] a $p \times p$ covariance matrix N times. Such models include MGARCH (Bollerslev et al., 1988) or multivariate stochastic volatility models (Harvey et al., 1994). These inversions (or Cholesky decompositions) would lead to a complexity of $\mathcal{O}(Nqp + Np^3)$ per likelihood evaluation. However, by working directly with the noisy $\hat{\boldsymbol{f}}$ instead of the

[1]In this context, "inverting" means decomposing (e.g., a Cholesky decomposition of) the matrix $\Sigma(x)$ in question. For instance, we may wish to take the Cholesky decomposition to take the determinant of $\Sigma^{-1}(x)$, or the matrix vector product $\boldsymbol{y}^\top(x)\Sigma^{-1}(x)\boldsymbol{y}(x)$.

noise free $\boldsymbol{f}$, evaluating the likelihood requires no costly or numerically unstable inversions, and thus has a complexity of only $\mathcal{O}(Nqp)$.

The computational complexity of variational Bayes is dominated by the $\mathcal{O}(N^3)$ inversions required to calculate the covariance of the node and weight functions in the E-step. Naively q and qp such inversions are required per iteration for the node and weight functions respectively, giving a total complexity of $\mathcal{O}(qpN^3)$. However, under VB the covariances of the weight functions for the same p are all equal, reducing the complexity to $\mathcal{O}(qN^3)$. If p is large the $\mathcal{O}(pqN^2)$ cost of calculating the weight function means may become significant. Although the per iteration cost of VB is actually higher than for MCMC, far fewer iterations are typically required to reach convergence.

We see that when fixing q and p, the computational complexity of GPRN scales cubically with the number of data points, like standard Gaussian process regression. On modern computers, this limits GPRN to datasets with fewer than about $N = 10000$ points. However, one could adopt a sparse representation of GPRN, for example following the DTC (Csató and Opper, 2001; Quiñonero-Candela and Rasmussen, 2005; Rasmussen and Williams, 2006; Seeger et al., 2003), PITC (Quiñonero-Candela and Rasmussen, 2005), or FITC (Snelson and Ghahramani, 2006) approximations, which would lead to $\mathcal{O}(M^2N)$ scaling where $M \ll N$. Importantly, if the input space $\mathcal{X}$ is on a grid, then the computational complexity for inference in N can reduce to $\mathcal{O}(N \log N)$ using Toeplitz methods (section 2.6.3.1), *with no loss of predictive accuracy.* For instance, these conditions for Toeplitz methods apply to essentially all time series applications, such as the multivariate volatility applications in this chapter. For inputs on a grid, one may also exploit Kronecker structure, for fast and exact inference and learning as in chapter 5 and Saatchi (2011).

Fixing q and N, the per iteration (for MCMC or VB) computational complexity of GPRN scales linearly with p. Overall, the computational demands of GPRN compare favourably to most multi-task GP models, which commonly have a complexity of $\mathcal{O}(p^3N^3)$ (e.g. SLFM, LMC, and CMOGP in the experiments) and do not account for either input dependent signal or noise correlations. Moreover, multivariate volatility models, which account for input dependent noise correlations, are commonly intractable for $p > 5$ (Engle, 2002; Gouriéroux et al., 2009). The 1000 dimensional gene expression experiment is tractable for GPRN, but intractable for the alternative multi-task models used on the 50 dimensional gene expression set, and the multivariate volatility models. Note that the computational efficiency gains in the GPRN come about through exploiting the structure of the GPRN during inference. Similar structure could also be exploited, for example, in the SLFM model, which can be viewed as a special case of the GPRN (section 3.5).

Using either MCMC or VB with the GPRN, the memory requirement is $\mathcal{O}(N^2)$ if all covariance kernels share the same hyperparameters, $\mathcal{O}(qN^2)$ is the node functions have different kernel hyperparameters, and $\mathcal{O}(pq^2N^2)$ if all GPs have different kernel hyperparameters. This memory requirement comes from storing the information in the block diagonal C_B in the prior $p(\boldsymbol{u}|\sigma_f, \boldsymbol{\theta}_f, \boldsymbol{\theta}_w)$. This memory requirement can also be significantly reduced at no loss in predictive accuracy if the relevant covariance matrices have Toeplitz or Kronecker structure (e.g., section 5.3 and chapter 5).

3.5 Related Work

Gaussian process regression networks are related to a large body of seemingly disparate work in machine learning, econometrics, geostatistics, physics, and probability theory.

In machine learning, the semiparametric latent factor model (SLFM) (Teh et al., 2005) was introduced to model multiple outputs with fixed signal correlations. SLFM specifies a linear mixing of latent Gaussian processes. The SLFM is similar to the linear model of coregionalisation (LMC) (Journel and Huijbregts, 1978) and intrinsic coregionalisation model (ICM) (Goovaerts, 1997) in geostatistics, but the SLFM incorporates important Gaussian process hyperparameters like length-scales, and methodology for learning these hyperparameters. The SLFM in Teh et al. (2005) also uses the efficient informative vector machine (IVM) framework

(Lawrence et al., 2003) for inference. In machine learning, the SLFM has also been developed as "Gaussian process factor analysis" (Yu et al., 2009), with an emphasis on time being the input (predictor) variable.

For changing correlations, the Wishart process (Bru, 1991) was first introduced in probability theory as a distribution over a collection of positive definite covariance matrices with Wishart marginals. It was defined as an outer product of autoregressive Gaussian processes restricted to a Brownian motion or Ornstein-Uhlenbeck covariance structure. In the geostatistics literature, Gelfand et al. (2004) applied a Wishart process as part of a linear coregionalisation model with spatially varying signal covariances, on a $p = 2$ dimensional real-estate example. Later Gouriéroux et al. (2009) returned to the Wishart process of Bru (1991) to model multivariate volatility, letting the noise covariance be specified as an outer product of AR(1) Gaussian processes, assuming that the covariance matrices $\Sigma(t) = \text{cov}(\boldsymbol{y}|t)$ are observables on an evenly spaced one dimensional grid. In machine learning, Wilson and Ghahramani (2010b, 2011) introduced the generalised Wishart process (GWP)[1], which generalises the Wishart process of (Bru, 1991) to a process over arbitrary positive definite matrices (Wishart marginals are not required) with a flexible covariance structure, and using the GWP, extended the GP framework to account for input dependent noise correlations (multivariate volatility), without assuming the noise is observable, or that the input space is 1D, or on a grid. Fox and Dunson (2011) also propose a related Bayesian nonparametric process over covariance matrices. More on the GWP is in section 3.8, with novel extensions in appendix B.

Gaussian process regression networks act as both a multi-task and multivariate volatility model. The GPRN and the GWP are closely related to the model of Gelfand et al. (2004). The GPRN (and also the GWP, in many respects) differs from Gelfand et al. (2004) in that 1) the GPRN incorporates and estimates Gaussian process hyperparameters, like length-scales, and thus learns more covariance structure from data data, 2) is tractable for $p > 3$ due to scalable inference (the VB and ESS inference procedures we present here are significantly more efficient than the Metropolis-Hastings proposals in Gelfand et al. (2004)), 3) is used as

[1]The relationship between the GWP and GPRN is discussed in more detail in section 3.8.

a latent factor model (where $q < p$), 4) incorporates an input dependent noise correlation model.

4) is especially pertinent, as a noise model in general strongly influences a regression on the signal, even if the noise and signal models are a priori independent. In the GPRN prior of Equation (3.3) the noise and signal correlations are explicitly related: through sharing $W(x)$, the signal and noise are encouraged to increase and decrease together (which provides an inductive bias which is *critical* for learning input dependent signal and noise structure using a model like the GPRN). The noise model is an example of a GWP, although with the inference presented in section 3.3, the GPRN scales linearly and not cubically with p (as in Wilson and Ghahramani (2010b)), per iteration of ESS or VB. This inference could be applied to the GWP for the same efficiency gains. If the GPRN is exposed solely to input dependent noise, the length-scales on the node functions $\boldsymbol{f}(x)$ will train to large values, turning the GPRN into solely a multivariate volatility model: all the modelling then takes place in $W(x)$. In other words, through learning Gaussian process hyperparameters, the GPRN can automatically vary between a multi-task and multivariate volatility model, making the GPRN a highly distinct model. The hyperparameters in the GPRN are also important for distinguishing between the behaviour of the weight and node functions. We may expect, for example, that the node functions will vary more quickly than the weight functions, so that the components of $\boldsymbol{y}(x)$ vary more quickly than the correlations between the components of $\boldsymbol{y}(x)$. The rate at which the node and weight functions vary is controlled by the Gaussian process length-scale hyperparameters, which are learned from data.

When $q = p = 1$, the GPRN resembles the nonstationary GP regression model of Adams and Stegle (2008). Likewise, when the weight functions are constants, the GPRN becomes the semiparametric latent factor model (SLFM) of Teh et al. (2005), except that the resulting GP regression network is less prone to over-fitting through its use of full Bayesian inference. The GPRN also somewhat resembles the natural sound model (MPAD) in section 5.3 of Turner (2010), except in MPAD the analogue of the node functions are AR(2) Gaussian processes, the inputs are 1D and on a regular grid, and the “weight functions” are a priori correlated and constrained to be positive.

Ver Hoef and Barry (1998) and Higdon (2002) in geostatistics and Boyle and Frean (2004) in machine learning proposed an alternate convolution GP model for multiple outputs (CMOGP) with fixed signal correlations, where each output at each $x \in \mathcal{X}$ is a mixture of latent Gaussian processes mixed across the whole input domain $\mathcal{X}$.

3.6 Experiments

We compare the GPRN to multi-task learning and multivariate volatility models. We also compare between variational Bayes (VB) and elliptical slice sampling (ESS) inference within the GPRN framework. In the multi-task setting, there are p dimensional observations $\boldsymbol{y}(x)$, and the goal is to use the correlations between the elements of $\boldsymbol{y}(x)$ to make better predictions of $\boldsymbol{y}(x_*)$, for a test input x_*, than if we were to treat the dimensions (aka "tasks") independently. A major difference between the GPRN and alternative multi-task models is that the GPRN accounts for signal correlations that *change* with x, and input dependent noise correlations, rather than fixed correlations. We compare to multi-task GP models on gene expression and geostatistics datasets.

To specifically test the GPRN's ability to model input dependent noise covariances (multivariate volatility), we compare predictions of $\text{cov}[\boldsymbol{y}(x)] = \Sigma(x)$ to those made by popular multivariate volatility models on benchmark financial datasets.

In all experiments, the GPRN uses squared exponential covariance functions, with a length-scale shared across all node functions, and another length-scale shared across all weight functions. The GPRN is robust to initialisation. We use an adversarial initialisation of $\mathcal{N}(0, 1)$ white noise for all Gaussian process functions.

3.6.1 Gene Expression

Tomancak et al. (2002) measured gene expression levels every hour for 12 hours during Drosophila embryogenesis; they then repeated this experiment for an independent replica (a second independent time series). This dataset was subsequently analyzed by Alvarez and Lawrence (2011). Gene expression is activated and deactivated by transcription factor proteins. We focus on genes which are thought to at least be regulated by the transcription factor `twi`, which influences mesoderm and muscle development in Drosophila (Zinzen et al., 2009). The assumption is that these gene expression levels are all correlated. We would like to use how these correlations change over time to make better predictions of time varying gene expression in the presence of transcription factors. In total there are 1621 genes (outputs) at $N = 12$ time points (inputs), on two independent replicas. For training, $p = 50$ random genes were selected from the first replica, and the corresponding 50 genes in the second replica were used for testing. We then repeated this experiment 10 times with a different set of genes each time, and averaged the results. We then repeated the whole experiment, but with $p = 1000$ genes. We used exactly the same training and testing sets as Alvarez and Lawrence (2011).

As in Alvarez and Lawrence (2011), we start with the $p = 50$ dataset, to compare with popular alternative multi-task methods (LMC, CMOGP, and SLFM) methods which have a complexity of $\mathcal{O}(N^3p^3)$ and do not scale to $p = 1000$ when using exact inference.[1,2]. On the $p = 1000$ dataset we compare to CMOFITC, CMODTC, and CMOPITC (Alvarez and Lawrence, 2011), sparse variants of CMOGP, which significantly outperforms LMC and SLFM on the $p = 50$ example. In both of these regressions, the GPRN is accounting for multivariate volatility; this is the first time a multivariate stochastic volatility model has been estimated for $p > 50$ (Chib et al., 2006). We assess performance using standardised mean square error (SMSE) and mean standardized log loss (MSLL). The SMSE is defined as

$$\text{SMSE} = \frac{||\boldsymbol{y}_* - \boldsymbol{f}_*||^2}{\sigma^2_{y*}}\,, \tag{3.18}$$

where $\boldsymbol{y}_*$ is the vector of test points, $\boldsymbol{f}_*$ is the vector of predictions at test locations, and $\sigma^2_{y_*}$ is the variance of the test points. The MSLL is the log likelihood of the data

[1]Here we are using the standard inference procedures for these alternative models. The SLFM, for instance, could scale with the same complexity as the GPRN if using the same inference procedures as the GPRN.

[2]We also implemented the SVLMC of Gelfand et al. (2004) but found it computationally intractable on the gene expression and geostatistics datasets, and on a subset of data it gave worse results than the other methods we compare to. SVLMC is not applicable to the multivariate volatility datasets.

minus the log likelihood obtained using the trivial model of using the empirical mean and covariance of the training instances to fit the data. The SMSE and MSLL metrics are similarly defined on page 23 of Rasmussen and Williams (2006). Smaller SMSE values and more negative MSLL values correpond to better fits of the data.

The results are in Table 3.1, under the headings `GENE` (50D) and `GENE` (1000D). For `SET 2` we reverse training and testing replicas in `SET 1`. The results for LMC, CMOGP, CMOFITC, CMOPITC, and CMODTC can also be found in Alvarez and Lawrence (2011). GPRN outperforms all of the other models, with between 46% and 68% of the SMSE, and similarly strong results on the MSLL error metric.[1] On both the 50 and 1000 dimensional datasets, the marginal likelihood for the network structure is sharply peaked at $q = 1$, as we might expect since there is likely one transcription factor `twi` controlling the expression levels of the genes in question.

Typical GPRN (VB) runtimes for the 50D and 1000D datasets were respectively 12 seconds and 330 seconds. These runtimes scale roughly linearly with dimension (p), which is what we expect. GPRN (VB) runs at about the same speed as the sparse CMOGP methods, and much faster than CMOGP, LMC and SLFM, which take days to run on the 1000D dataset. The GPRN (ESS) runtimes for the 50D and 1000D datasets were 40 seconds and 9000 seconds (2.5 hr), and required respectively 6000 and 10^4 samples to reach convergence, as assessed by trace plots of sample likelihoods. In terms of both speed and accuracy GPRN (ESS) outperforms all methods except GPRN (VB). GPRN (ESS) does not mix as well in high dimensions, and the number of ESS iterations required to reach convergence noticeably grows with p. However, ESS is still tractable and performing relatively well in $p = 1000$ dimensions, in terms of speed and predictive accuracy. Runtimes are on a 2.3 GHz Intel i5 Duo Core processor.

[1]Independent GPs severely overfit on `GENE`, giving an MSLL of ∞.

3.6.2 Jura Geostatistics

Here we are interested in predicting concentrations of cadmium at 100 locations within a 14.5 km^2 region of the Swiss Jura. For training, we have access to measurements of cadmium at 259 neighbouring locations. We also have access to nickel and zinc concentrations at these 259 locations, as well as at the 100 locations we wish to predict cadmium. While a standard Gaussian process regression model would only be able to make use of the cadmium training measurements, a multi-task method can use the correlated nickel and zinc measurements to enhance predictions. With the GPRN we can also make use of how the correlations between nickel, zinc, and cadmium change with location to further enhance predictions.

The network structure with the highest marginal likelihood has $q = 2$ latent node functions. The node and weight functions learnt using VB for this setting are shown in Figure 3.2. Since there are $p = 3$ output dimensions, the result $q < p$ suggests that heavy metal concentrations in the Swiss Jura are correlated. Indeed, using our model we can observe the *spatially varying* correlations between heavy metal concentrations, as shown for the noise correlations between cadmium and zinc in Figure 3.3. Although the correlation between cadmium and zinc is generally positive (with values around 0.6), there is a region where the correlations noticeably decrease, perhaps corresponding to a geological structure. The quantitative results in Table 3.1 suggest that the ability of the GPRN to learn these spatially varying correlations is beneficial for predicting cadmium concentrations.

We assess performance quantitatively using mean absolute error (MAE) between the predicted and true cadmium concentrations. We restart the experiment 10 times with different initialisations of the parameters, and average the MAE. The results are marked by `JURA` in Table 3.1. The experimental setup follows Goovaerts (1997) and Alvarez and Lawrence (2011). We found log transforming and normalising each dimension to have zero mean and unit variance to be beneficial due to the skewed distribution of the y-values (but we also include results on untransformed data, marked with *). All the multiple output methods give lower MAE than using an independent GP, and the GPRN outperforms SLFM and the other methods.

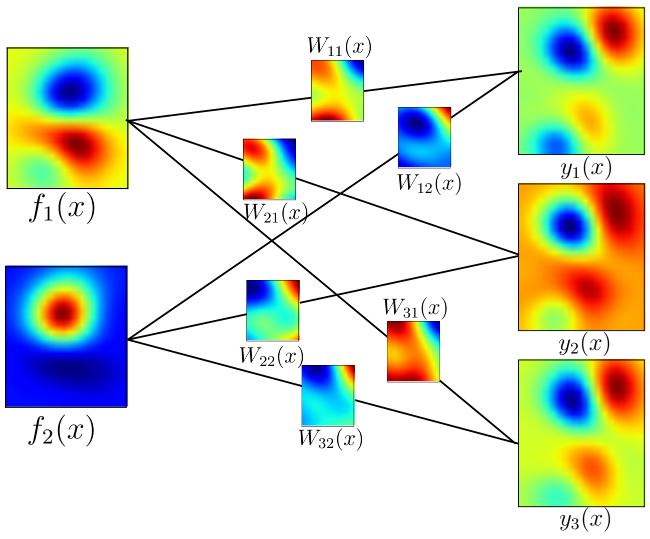


Figure 3.2: Network structure for the Jura dataset learnt by the GPRN (VB). The spatially varying node and weight functions shown, along with the predictive means for the observations. The three output dimensions are cadmium, nickel and zinc concentrations respectively. The horizontal and vertical axes are latitude and longitude as in Figure 3.3.

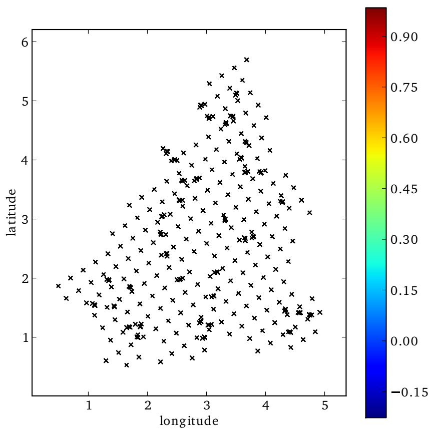


Figure 3.3: Spatially dependent correlations between cadmium and zinc learned by the GPRN. Markers show the locations where measurements were made.

For the JURA dataset, the improved performance of the GPRN is at the cost of a slightly greater runtime. However, the GPRN is accounting for input dependent signal and noise correlations, unlike the other methods. Moreover, the complexity of the GPRN scales linearly with p (per iteration), unlike the other methods which scale as $\mathcal{O}(N^3p^3)$. This scaling is why the GPRN runs relatively quickly on the 1000 dimensional gene expression dataset, for which the other methods are intractable. These data are available from `http://www.ai-geostats.org/`.

3.6.3 Multivariate Volatility

In the previous experiments the GPRN implicitly accounted for multivariate volatility (input dependent noise covariances) in making predictions of $\boldsymbol{y}(x_*)$. We now test the GPRN explicitly as a model of multivariate volatility, and assess predictions of $\Sigma(t) = \text{cov}[\boldsymbol{y}(t)]$. We make 200 historical predictions of $\Sigma(t)$ at observed time points, and 200 one day ahead forecasts. Historical predictions can be used, for example, to understand a past financial crisis. The forecasts are assessed using the log likelihood of new observations under the predicted covariance, denoted $\mathcal{L}$ Forecast. We follow Wilson and Ghahramani (2010b), and predict $\Sigma(t)$ for returns on three currency exchanges (EXCHANGE) and five equity indices (EQUITY) processed as in Wilson and Ghahramani (2010b). These datasets are especially suited to MGARCH, the most popular multivariate volatility model, and have become a benchmark for assessing GARCH models (Brooks et al., 2001; Brownlees et al., 2009; Hansen and Lunde, 2005; McCullough and Renfro, 1998; Poon and Granger, 2005). We compare to full BEKK MGARCH (Engle and Kroner, 1995), the generalised Wishart process (Wilson and Ghahramani, 2010b), and the original Wishart process (Bru, 1991; Gouriéroux et al., 2009).

We see in Table 3.1 that GPRN (ESS) is often outperformed by GPRN (VB) on multivariate volatility sets, suggesting convergence difficulties with ESS. The high historical MSE for GPRN on EXCHANGE is essentially training error, and less meaningful than the encouraging step ahead forecast likelihoods; to harmonize with the econometrics literature, historical MSE for EXCHANGE is between the learnt covariance $\Sigma(x)$ and observed $\mathbf{y}(x)\mathbf{y}(x)^\top$. Wilson and Ghahramani (2010b) contains further details. The small differences in performance between the GPRN and GWP on these datsets are a result of minor differences in implementation and the fact we used GPRN as a factor model with $q < p$ which is suitable for these particular datasets.

Overall, the GPRN shows promise as both a multi-task and multivariate volatility model, especially since the multivariate volatility datasets are suited to MGARCH. These data were obtained using Datastream (`http://www.datastream.com/`).

3.7 Discussion

A Gaussian process regression network (GPRN) has a simple and interpretable structure, and generalises many of the recent extensions to the Gaussian process regression framework. The model naturally accommodates input dependent signal and noise correlations between multiple output variables, heavy tailed predictive distributions, input dependent length-scales and amplitudes, and adaptive covariance functions. Furthermore, GPRN has scalable inference procedures, and strong empirical performance on several real datasets. In the future, one could apply the ideas behind GPRN to classification, so that one can model non-mutually exclusive class labels with predictor dependent correlations. For example, the class labels could be education level and profession, and the predictor could be geographical location.

In the next sections we describe the relationship between the GPRN and the generalised Wishart process (GWP), and provide some more detail about generalised Wishart processes. We also generalise the GPRN framework to adaptive networks, and describe applications in nuclear magnetic spectroscopy, ensemble learning, and changepoint modelling.

In the next chapter we introduce closed form kernels for automatic pattern discovery and extrapolation. These kernels are highly complementary with the GPRN framework – and can be used as the node kernels in a GPRN to form a powerful non-stationary process, which we describe in section 4.4.

Table 3.1: Comparative performance on all datasets.

GENE (50D)	Average SMSE	Average MSLL
SET 1:		
GPRN (VB)	0.3356 ± 0.0294	$\mathbf{-0.5945 \pm 0.0536}$
GPRN (ESS)	$\mathbf{0.3236 \pm 0.0311}$	-0.5523 ± 0.0478
LMC	0.6069 ± 0.0294	-0.2687 ± 0.0594
CMOGP	0.4859 ± 0.0387	-0.3617 ± 0.0511
SLFM	0.6435 ± 0.0657	-0.2376 ± 0.0456
SET 2:		
GPRN (VB)	0.3403 ± 0.0339	$\mathbf{-0.6142 \pm 0.0557}$
GPRN (ESS)	$\mathbf{0.3266 \pm 0.0321}$	-0.5683 ± 0.0542
LMC	0.6194 ± 0.0447	-0.2360 ± 0.0696
CMOGP	0.4615 ± 0.0626	-0.3811 ± 0.0748
SLFM	0.6264 ± 0.0610	-0.2528 ± 0.0453
GENE (1000D)	Average SMSE	Average MSLL
SET 1:		
GPRN (VB)	$\mathbf{0.3473 \pm 0.0062}$	$\mathbf{-0.6209 \pm 0.0085}$
GPRN (ESS)	0.4520 ± 0.0079	-0.4712 ± 0.0327
CMOFITC	0.5469 ± 0.0125	-0.3124 ± 0.0200
CMOPITC	0.5537 ± 0.0136	-0.3162 ± 0.0206
CMODTC	0.5421 ± 0.0085	-0.2493 ± 0.0183
SET 2:		
GPRN (VB)	$\mathbf{0.3287 \pm 0.0050}$	$\mathbf{-0.6430 \pm 0.0071}$
GPRN (ESS)	0.4140 ± 0.0078	-0.4787 ± 0.0315
CMOFITC	0.5565 ± 0.0425	-0.3024 ± 0.0294
CMOPITC	0.5713 ± 0.0794	-0.3128 ± 0.0138
CMODTC	0.5454 ± 0.0173	0.6499 ± 0.7961
JURA	Average MAE	Training (secs)
GPRN (VB)	$\mathbf{0.4040 \pm 0.0006}$	1040
GPRN* (VB)	0.4525 ± 0.0036	1190
SLFM (VB)	0.4247 ± 0.0004	614
SLFM* (VB)	0.4679 ± 0.0030	810
SLFM	0.4578 ± 0.0025	792
Co-kriging	0.51	
ICM	0.4608 ± 0.0025	507
CMOGP	0.4552 ± 0.0013	784
GP	0.5739 ± 0.0003	74
EXCHANGE	Historical MSE	$\mathcal{L}$ Forecast
GPRN (VB)	3.83×10^{-8}	**2073**
GPRN (ESS)	6.120×10^{-9}	2012
GWP	$\mathbf{3.88 \times 10^{-9}}$	2020
WP	$\mathbf{3.88 \times 10^{-9}}$	1950
MGARCH	3.96×10^{-9}	2050
EQUITY	Historical MSE	$\mathcal{L}$ Forecast
GPRN (VB)	0.978×10^{-9}	2740
GPRN (ESS)	$\mathbf{0.827 \times 10^{-9}}$	2630
GWP	2.80×10^{-9}	**2930**
WP	3.96×10^{-9}	1710
MGARCH	6.68×10^{-9}	2760

3.8 Relationship with Generalised Wishart Processes

In section 3.2, we noted that the noise covariance of a GPRN,

$$\Sigma(x) = \sigma_f^2 W(x)W(x)^\top + \sigma_y^2 I_p \,, \tag{3.19}$$

is an example of a *generalised Wishart process* (Wilson and Ghahramani, 2010b, 2011), because the entries of $W(x)$ are GPs. Here we describe generalised Wishart processes in some more detail. In the process, the connections between GPRNs and GWPs will become increasingly clear.

Gaussian processes are distributions over functions. Intuitively we can think of these functions as sequences of indexed (e.g., input dependent) points, which are marginally Gaussian and correlated with one another as described by a covariance kernel. We can therefore intuitively think of a sequence of input dependent matrices as a generalised function. A generalised Wishart process (GWP) is a distribution over sequences of matrices which are correlated using a kernel function.[1] In the simplest formulation of the GWP, these matrices are marginally Wishart distributed. Therefore the GWP can be thought of as a generalisation of the GP framework to sequences of matrices.

To construct a GWP with Wishart marginals, we start with νp independent Gaussian process functions, $u_{id}(x) \sim \mathcal{GP}(0, k)$, where $i = 1, \ldots, \nu$ and $d = 1, \ldots, p$. This means

$$\mathrm{cov}[u_{id}(x), u_{i'd'}(x')] = k(x, x')\delta_{ii'}\delta_{dd'} \,, \tag{3.20}$$

$$(u_{id}(x_1), u_{id}(x_2), \ldots, u_{id}(x_N))^\top \sim \mathcal{N}(0, K) \,, \tag{3.21}$$

where δ_{ij} is the Kronecker delta, and K is an $N \times N$ covariance matrix with elements $K_{ij} = k(x_i, x_j)$. Let $\hat{\boldsymbol{u}}_i(x) = (u_{i1}(x), \ldots, u_{ip}(x))^\top$, and let L be the lower Cholesky decomposition of a $p \times p$ scale matrix V, such that $LL^\top = V$. Also,

[1]See also Fox and Dunson (2011) and Gelfand et al. (2004) for a related approach.

suppose, without loss of generality, that $k(x,x) = 1$. We can then construct a GWP as

$$\Sigma(x) = \sum_{i=1}^{\nu} L\hat{\boldsymbol{u}}_i(x)\hat{\boldsymbol{u}}_i^{\top}(x)L^{\top}\,. \tag{3.22}$$

Theorem 3.8.1. *At each $x \in \mathcal{X}$ the covariance matrix $\Sigma(x)$ defined in* (3.22) *has a Wishart marginal distribution $\mathcal{W}_p(V,\nu)$.*

Proof 3.8.1. *Each element of the vector $\hat{\boldsymbol{u}}_i(x)$ is a univariate Gaussian with zero mean and variance $k(x,x) = 1$. Since these elements are uncorrelated, $\hat{\boldsymbol{u}}_i(x) \sim \mathcal{N}(\mathbf{0}, I)$. Therefore $L\hat{\boldsymbol{u}}_i(x) \sim \mathcal{N}(0,V)$, since $\mathbb{E}[L\hat{\boldsymbol{u}}_i(x)\hat{\boldsymbol{u}}_i(x)^{\top}L^{\top}] = LIL^{\top} = LL^{\top} = V$. Eq.* (3.22) *is a sum of outer products of independent $\mathcal{N}(0,V)$ random variables, and there are ν terms in the sum, so by definition this has a Wishart distribution $\mathcal{W}_p(V,\nu)$. It was not a restriction to assume $k(x,x) = 1$, since any scaling of k (e.g. ak with $a \in \mathbb{R}$) can be absorbed into the matrix L.*

We can re-write this construction in a way that lends itself to modification. If $A(x)$ is a $p \times \nu$ matrix of GPs, such that $A_{ij}(x) \sim \mathcal{GP}(0,k)$, then

$$\Sigma(x) = LA(x)A(x)^{\top}L^{\top}\,, \tag{3.23}$$

is equivalent to Equation (3.22). A simple modification (for parsimony) is to let $A(x)$ be lower triangular, at the expense of expressive power. Eq. (3.19) is a factor representation of a GWP, since $W(x)$ is $p \times q$, typically with $q \ll p$. In appendix B3 we construct a GWP, with lower triangular $A(x)$, which has more expressive power than (3.22), since it has a real valued degrees of freedom ν. This construction makes use of *copula processes* (Wilson and Ghahramani, 2010a).

Definition 3.8.1. *A Generalised Wishart Process is a collection of positive semi-definite random matrices indexed by $x \in \mathcal{X}$ and constructed from outer products of points from collections of stochastic processes like in Eqs.* (3.22) *or* (3.23). *If the random matrices have 1) Wishart marginal distributions, meaning that $\Sigma(x) \sim W_p(V,\nu)$ at every $x \in \mathcal{X}$, and 2) dependence on x as defined by a kernel $k(x,x')$, then we write*

$$\Sigma(x) \sim \mathcal{GWP}(V,\nu,k(x,x'))\,. \tag{3.24}$$

The $\mathcal{GWP}$ notation of Definition 3.8.1 is just like the notation for a Wishart distribution, but includes a kernel which controls the dynamics of how $\Sigma(x)$ varies

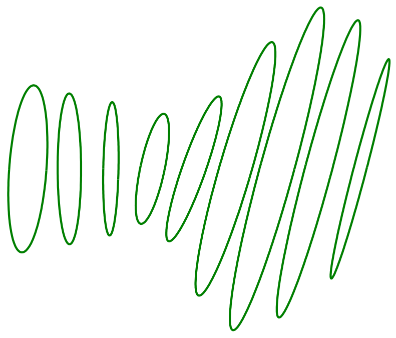


Figure 3.4: A draw from a generalised Wishart process (GWP). Each ellipse is represented by a 2×2 covariance matrix indexed by time, which increases from left to right. The rotation indicates the correlation between the two variables, and the major and minor axes scale with the eigenvalues of the matrix. The way in which these matrices vary with time is controlled by the kernel function k. Like a draw from a Gaussian process is a collection of function values indexed by time, a draw from a GWP is a collection of matrices indexed by time.

with x. This notation pleasingly compartmentalises the shape parameters and temporal (or spatial) dynamics parameters. We show an example of these dynamics in Figure 3.4 in a draw from a GWP. In appendix B1 we explicitly show how $\text{cov}[\Sigma_{ij}(x), \Sigma_{ls}(x')]$ is controlled by $k(x, x')$.

Definition 3.8.1 contains the construction of Equation (3.22) as a special case, since Gaussian processes need not be used. In appendix B4, for example, we construct a GWP with copula processes (Wilson and Ghahramani, 2010a) instead of Gaussian processes, resulting in marginals which belong to a more general class of matrix variate distributions.

Like the Gaussian process, the generalised Wishart process is a fundamental model, with limitless applications. However, GWPs appear to have obvious promise in econometrics, for modelling multivariate volatility. The generalised Wishart process, associated inference procedures, and an application to multivariate volatility, are discussed in more depth in Wilson and Ghahramani (2010b, 2011). Novel properties and constructions of the GWP are presented in appendix B.

We now present *adaptive networks*, a generalisation of the GPRN framework.

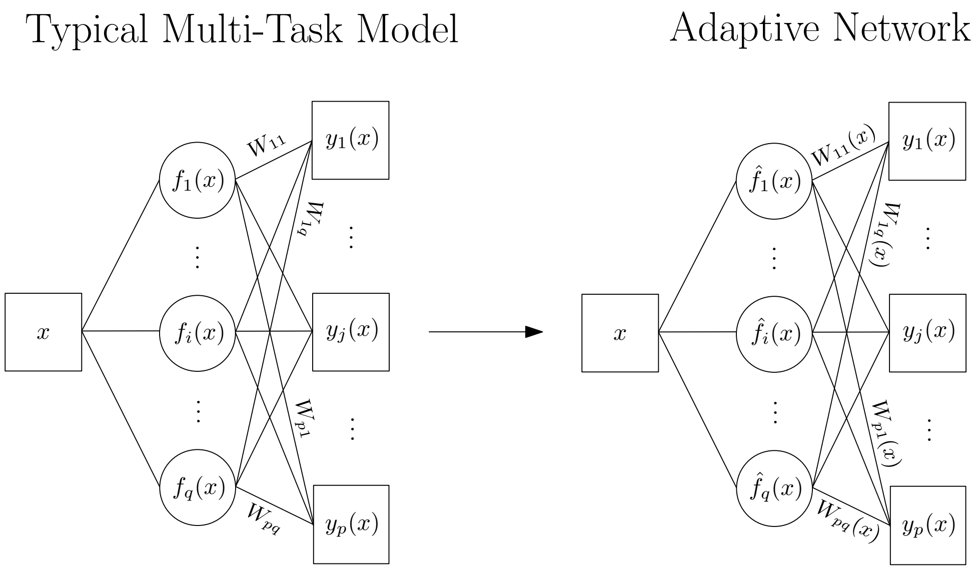


Figure 3.5: Transformation from a typical multi-task network (e.g. a neural network, or the semiparametric latent factor model (Teh et al., 2005)) to an adaptive network. For a review of the depicted adaptive network see section 3.2.

3.9 Generalisation to Adaptive Networks

Underlying the Gaussian process regression network is the high level idea that connections in undirected graphical models, typically used to represent neural networks, can be functions rather than constants. In this section we wish to emphasize that many of the properties of the GPRN do not depend on using Gaussian processes at all. Gaussian processes are one of many convenient choices for the node and weight functions. We will refer to a network with *input dependent* weights and nodes (as in Figure 3.5b) – regardless of what precise form these weight and node functions might take – as an *adaptive network*.

In a typical multi-task network structure there is an input (predictor) layer, x, which represents times, spatial locations, or any other predictor variables. Depending on the inputs is a layer of hidden basis functions (nodes) $f_1, \dots, f_q$. Connected to this layer of basis functions are the response variables $y_1, \dots, y_p$. These response variables, for example, could be the observable noisy expression levels of p different genes. The p genes are correlated because they share basis functions through the connections W in the graph. In an adaptive network, the connections become functions of the inputs: $W \to W(x)$. The transformation from a typical network structure to an adaptive network structure is shown in Figure 3.5. This structure naturally allows for a great amount of flexibility without sacrificing tractability. In particular,

- There are now signal correlations between the responses that vary with inputs. For example, when predicting time-varying gene expression levels, we can learn how the correlations between expression levels vary with time, and use that extra information to make better predictions.
- If we place additive Gaussian noise on the nodes, so that $f_1, \dots, f_q \to \hat{f}_1, \dots, \hat{f}_q$, then we automatically have an input dependent noise (multivariate volatility) covariance model for the responses, because of how the additive noise is mixed by the weight functions to form the responses. Typically a model of input dependent noise has a complexity that scales at least cubically with the number of responses. In the case of the adaptive network, the scaling can be linear in the number of responses, as explained in section 3.3 on GPRN inference. In an adaptive network, the magnitude of the signal and noise are encouraged (but not restricted) to increase together, as discussed in section 3.5. I have found that encouraging but not restricting the magnitude of the signal and noise to increase and decrease together is generally a good assumption and will provide superior predictions to models which treat the signal and noise as a priori independent.
- If the node functions are Gaussian processes with different covariance kernels, then the covariance kernel for each response is an input dependent mixture of kernels. Such an expressive covariance kernel allows for significant *kernel-learning*. One can have for example, non-parametric non-stationary *length-scales*. Overall, the covariance kernel[1] for each response can significantly adapt to data, unlike the typical squared exponential covariance kernels (Rasmussen and Williams, 2006).
- If the node functions are entirely different regression methods – for instance, one node function is a linear regression model, another is a neural network

[1]See section 2.4 for a discussion of covariance kernels.

model, etc. – then the adaptive network can be viewed as an *ensemble learning* method.[1] Many different regressors are then combined to make predictions of the responses. Each regressor has a different weighting which depends on the input space. At certain input locations, one regressor may have more value than the others, and will receive a greater weighting. This model would generalise the mixtures of experts framework (Jacobs et al., 1991).

- The concept of an adaptive network is independent of Gaussian processes or Bayesian inference.

The Gaussian process regression network (GPRN) demonstrates the promise of an adaptive network structure: the GPRN has impressive predictive performance on several scientific datasets. In the next sections I briefly propose models for NMR spectroscopy and change points, to exemplify the variety of applications possible with adaptive network structures. These are not fully developed applications, but rather pointers to potentially interesting future work.

3.9.1 NMR Spectroscopy Application

Here we briefy propose a model for NMR spectroscopy, as an example application of an adaptive network structure.

Nuclear magnetic resonance (NMR) spectroscopy exploits the magnetic properties of atomic nuclei to discover the structure, dynamics, reaction state and chemical environment of molecules. NMR spectroscopy has been instrumental in understanding molecular properties, developing pharmaceuticals, and in medicine and chemical engineering in general. Robert Ernst won the Nobel Prize in chemistry for Fourier transform spectroscopy (Ernst, 1992), the most widely used NMR spectroscopy technique.

To understand how NMR spectroscopy works at a high level, imagine a chemical mixture is placed in a strong magnetic field. Nuclei within this mixture that

[1]chapter 14 of Bishop (2006) contains an introduction to boosting and other ensemble learning techniques for combining models.

have non-zero spin will interact with the magnetic field to produce “magnetic moments”. We can imagine these magnetic moments as small bar magnets. Radio frequency (rf) pulses are then directed at the mixture, exerting a torque on these “bar magnets”, causing them to precess perpendicular to the strong magnetic field. The rotating magnets create a magnetic flux through a coil in the NMR machine, which induces a current with periodic components at the resonant frequencies of the chemical mixture. The voltage associated with this current is what is measured in an NMR experiment and is called the free induction decay (FID).[1] The resonant frequencies are sensitive to the local molecular structure, which is what permits NMR to be used as a spectroscopic technique.

Conventional Fourier transform spectroscopy does not explicitly model decay or noise in the signal, and is thus limited to studying systems with relatively high signal to noise ratios. Moreover, it is assumed in conventional Fourier transform spectroscopy that the system in question is not changing – for example, that chemicals in a mixture are not reacting over the acquisition time of the FID signal.

Systems which are transient over the acquisition time of an FID signal (acquisition time is at least 10-100 ms using the most recent hardware developments such as stopped flow (Hore et al., 1997) or rapid injection (Bowen and Hilty, 2010)) are of interest in chemical engineering and biochemistry. These systems are widely encountered in rapid chemical reactions (Kühne et al., 1969), enzyme catalysis (Grimaldi and Sykes, 1975), and protein and nucleic acid folding (Balbach et al., 1995; Fürtig et al., 2007).

Here we propose an NMR spectroscopy model which accounts for decay, noise, and prior information about resonant frequencies in the FID signal. Moreover, this model – an example of an adaptive network – is rather uniquely capable of accounting for transient FID signals, with a high degree of flexibility. For concreteness, we imagine we are modelling chemical mixtures – but the same exact model could be applied to other transient FID signals.

[1]The FID signal is usually recorded in two channels, a real and imaginary channel, which are close to 90 degrees out of phase. For clarity, we consider only the real channel here, because accounting for the imaginary channel adds significant notational clutter and does not change any of the fundamental modelling concepts.

We model the FID, $y(t)$, as

$$\begin{aligned} y(t) = A_1(t)\sum_{i=1}^{m_1} B_i^{(1)}\cos((\omega_i^{(1)} - \omega_0)(t+\tau)+\theta)e^{-\alpha t} \\ + A_2(t)\sum_{i=1}^{m_2} B_i^{(2)}\cos((\omega_i^{(2)} - \omega_0)(t+\tau)+\theta)e^{-\alpha t} \\ + \ldots + A_r(t)\sum_{i=1}^{m_r} B_i^{(r)}\cos((\omega_i^{(r)} - \omega_0)(t+\tau+\theta))e^{-\alpha t} \\ + \epsilon(t)\,, \end{aligned} \tag{3.25}$$

for r chemical species, with amplitude functions $A_j(t) \sim \mathcal{GP}(0, k_i)$, resonant frequencies $\{\omega\}_i^{(j)}$, global phase θ and time delay τ, intensities $\{B\}_i^{(j)}$, reference frequency ω_0, decay constant α, and noise $\epsilon(t) \sim \mathcal{N}(0, v)$. Chemical r has m_r resonant frequencies. We assume r is known and fixed. The relative values $|A_i(t)/A_j(t)|$ represent the relative concentrations of chemicals i and j at time t. Since we are interested in $|A_i(t)/A_j(t)|$, the sign of the amplitudes is not important. Moreover, the model of Eq. (3.25) can be generalised to have local phase variables $\{\theta\}_i^{(j)}$ which can absorb the sign of $A_j(t)$, if desired. Various statistical models of the FID have been proposed, resembling the stationary form of the model in Eq. (3.25) (where $A(t) \to A$) (Bretthorst, 1990; Rubtsov and Griffin, 2007; Wilson et al., 2014). Hutton et al. (2009) develop a specialised model for mixtures of water and metabolites, where the amplitude coefficient representing water is time-varying. Eq. (3.25) also resembles natural sound models for amplitude modulation, such as Turner and Sahani (2007), which use AR processes (appendix A), and the model of Qi et al. (2002) which use AR processes for amplitude modulation, for unevenly sampled time series data.

We can assume ω_0 and $\{B\}_i^{(j)}$ are fixed: ω_0 can be calibrated, and $\{B\}_i^{(j)}$ can be fixed from the tabulated values in `http://sdbs.riodb.aist.go.up`. In preliminary experiments (Wilson et al., 2014), we found it critical to the success of the proposed model to finely estimate the resonant frequencies from data – it is not sufficient to use their approximate tabulated values. However, it is sensible to leverage prior information about their approximate values through a prior distribution over ω.

We rewrite Eq. (3.25) as

$$y(t) = A_1(t)\phi_1(t,\psi_1) + A_2(t)\phi_2(t,\psi_2) + \cdots + A_r(t)\phi_r(t,\psi_r) + \epsilon(t) \tag{3.26}$$

$$A_i(t) \sim \mathcal{GP}(m_i(t), k_i(t,t'))\,, \tag{3.27}$$

$$\epsilon(t) \sim \mathcal{N}(0, v)\,, \tag{3.28}$$

where ψ_j represents all variables associated with chemical j, such as the resonant frequencies $\{\omega\}_i^{(j)}$, except the kernel hyperparameters for the Gaussian process $A_j(t)$. For notational simplicity, I will not explicitly include ψ in subsequent equations.

Since we have placed a Gaussian process over each amplitude $A_i(t)$ (all of which are latent functions), the FID is a linear Gaussian system, and we can thus infer an analytic posterior distribution over each $A_i(t)$ given observations $\boldsymbol{y} = [y(t_1), \ldots, y(t_N)]$, allowing us to learn the history and forecast the future of chemical reactions, from a signal acquired during a reaction.

It is promising that we can perform exact Bayesian inference over the amplitude functions – the main quantity of interest – with the highly flexible nonparametric model in (3.26)-(3.27). This model will have support for a wide range of amplitude functions, but also has the flexibility to concentrate that support on a physical model. This model is an example of an *adaptive network*. We can view the amplitude functions as the latent basis functions, and the connections from the amplitudes to the response (the FID signal) , the *weight-functions* in chapter 3, as the non-linear basis functions $\phi_1(t), \ldots, \phi_r(t)$. The adaptive network in (3.26) is shown in Figure 3.6, in analogy with Figure 3.1 for the Gaussian process regression network.

In order to predict the progress of the chemical reaction at any time t_*, given N observations of the FID at times $(t_1, t_2, \ldots, t_N)$, we wish to find $p(\boldsymbol{a}_*|\boldsymbol{y})$, where $\boldsymbol{y}$ and $\boldsymbol{a}_*$ are defined as

$$\boldsymbol{y} = [\sum_{j=1}^{r} A_j(t_1)\phi_j(t_1), \ldots, \sum_{j=1}^{r} A_j(t_N)\phi_j(t_N)] + \boldsymbol{\epsilon}\,, \tag{3.29}$$

$$\boldsymbol{a}_* = [A_1(t_*), A_2(t_*), \ldots, A_r(t_*)]\,, \tag{3.30}$$

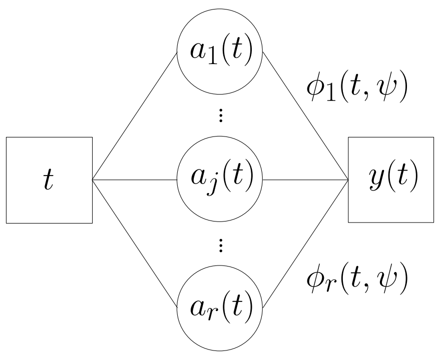


Figure 3.6: An adaptive network for modelling the NMR signal acquired during a time-varying reaction. The hidden layer of amplitude functions describe the relative concentrations of each chemical. There is a single response – the observed noisy free induction decay (FID) signal. The connections from the amplitude functions to the FID response are the basis functions $\phi_1(t), \ldots, \phi_r(t)$.

where $\boldsymbol{\epsilon} = (\epsilon(t_1), \ldots, \epsilon(t_N))^\top \sim \mathcal{N}(0, \sigma^2 I)$.

Fortunately, $y(t)$ is a GP with covariance function

$$k_y(t,t') = \phi_1(t)k_1(t,t')\phi_1(t') + \phi_2(t)k_2(t,t')\phi_2(t') + \cdots + \phi_r(t)k_r(t,t')\phi_r(t')\,. \tag{3.31}$$

The joint distribution over $\boldsymbol{y}$ and $\boldsymbol{a}_*$ is[1]

$$\begin{bmatrix} \boldsymbol{y} \\ \boldsymbol{a}_* \end{bmatrix} \sim \mathcal{N}(\begin{bmatrix} \boldsymbol{m}_y \\ \boldsymbol{m}_{a_*} \end{bmatrix}, \begin{bmatrix} \overbrace{K_y}^{N\times N} + \sigma^2 I & \overbrace{K_{y,a_*}}^{N\times r} \\ \overbrace{K_{a_*,y}}^{r\times N} & \overbrace{K_{a_*,a_*}}^{r\times r} \end{bmatrix}) \tag{3.32}$$

with

$$K_y(i,j) = \phi_1(t_i)k_1(t_i,t_j)\phi_1(t_j) + \phi_2(t_i)k_2(t_i,t_j)\phi_2(t_j) + \cdots + \phi_r(t_i)k_r(t_i,t_j)\phi_r(t_j) \tag{3.33}$$

$$K_{a_*,y}(i,j) = \phi_i(t_j)k_i(t_j,t_*) \tag{3.34}$$

$$K_{a_*,a_*}(i,j) = \delta_{ij}\,, \tag{3.35}$$

where, for instance, $K_{a_*,y}(i,j)$ is row i and column j of $K_{a_*,y}$ in Eq. (3.32).

[1]For generality, we have allowed for deterministic mean functions $\boldsymbol{m}_y$ and $\boldsymbol{m}_{a_*}$.

From this joint distribution, we find that the posterior over $\boldsymbol{a}_*$ is

$$\boldsymbol{a}_*|\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, C) \tag{3.36}$$

$$\boldsymbol{\mu} = \boldsymbol{m}_{a*} + K_{a*,y}[K_y + \sigma^2 I]^{-1}(\boldsymbol{y} - \boldsymbol{m}_y) \tag{3.37}$$

$$C = K_{a*,a*} - K_{a*,y}[K_y + \sigma^2 I]^{-1}K_{a*,y}^{\top} \,. \tag{3.38}$$

The computational complexity for evaluating Eq. (3.36) is naively $\mathcal{O}(N^3)$, where N is the number of datapoints. Typically an N^3 complexity is not an issue in these NMR applications, where we would not expect to collect more than 10000 datapoints. However, we can exploit the structure of the covariance functions we use for exact $\mathcal{O}(N \log N)$ inference, as described in section 2.6.3.1.

The marginal likelihood of the data $\boldsymbol{y}$, having integrated away all amplitude functions, is given by Eq. (2.37) in chapter 2. One can use this marginal likelihood to estimate or infer distributions over all remaining parameters ψ – the parameters of the basis functions, the noise variance, and the kernel hyperparameters. However, we have found in Wilson et al. (2014) that the marginal likelihood is highly multimodal as a function of the frequency parameters $\{\omega\}_i^{(j)}$, and the frequencies are highly dependent on one another and the phase variables. Therefore one cannot use conventional sampling techniques (e.g. Metropolis-Hastings, Slice Sampling, Hamiltonian Monte Carlo, etc.) or greedy optimization methods (e.g. quasi-Newton, conjugate-gradients, steepest descent, Newton) to estimate ψ. To estimate ψ, we have had success with the SIMPSA algorithm (Cardoso et al., 1996), which combines simulated annealing with the non-linear simplex algorithm of Nelder and Mead (1965), a deterministic algorithm for global optimization. SIMPSA is particularly effective for continuous constrained global optimization, and in our preliminary work (Wilson et al., 2014) we have found it far more effective for NMR spectroscopy than greedy optimization with constraints.

Simulated annealing (Kirkpatrick et al., 1983) was inspired by annealing in metallurgy, where crystals are heated and cooled in a controlled manner in search of a minimum energy configuration. Unlike a "greedy" minimization algorithm, which will always move parameters so that the objective function (in this case the negative log marginal likelihood) decreases on every move, simulated annealing allows "uphill" moves (with a probability proportional to the temperature), so that the parameters do not get stuck in a local minimum. To move from an initial value, simulated annealing samples new values of ψ from a proposal distribution. These new values will be accepted with a probability proporational to the difference between the objective function evaluated at the new and old values, and the temperature. Initially, when the temperature is high, the moves are almost random, and as the temperature cools down, the moves become increasingly likely to go downhill (the algorithm becomes increasingly like a deterministic greedy optimisation algorithm).

3.9.2 Change Point Kernel

Adaptive regression networks – and GPRNs – can naturally be developed into powerful changepoint models. In an adaptive network, a univariate output variable $y(x)$ is an input dependent mixture of functions:

$$y(x) = w_1(x)f_1(x) + w_2(x)f_2(x) + \cdots + w_q(x)f_q(x)\,. \tag{3.39}$$

If the node functions $f_1(x), \ldots, f_q(x)$ have different covariance functions – or covariance functions with different hyperparameters – then $y(x)$ will switch between different covariance regimes. We can imagine if f_1 has a squared exponential (SE) kernel, f_2 has an Ornstein-Uhlenbeck (OU) kernel, and f_3 has a periodic kernel, $y(x)$ could switch between regions with smooth, OU, and periodic covariance structure, or some mixtures of these structures.

The changes in covariance structure can be made more discrete by warping the weight functions through sigmoid functions:

$$y(x) = \sigma(w_1(x))f_1(x) + \cdots + \sigma(w_q(x))f_q(x)\,. \tag{3.40}$$

If we consider two node functions, and wish σ to act as a switch between the two functions, we can adapt the model to

$$y(x) = \sigma(w(x))f_1(x) + \sigma(-w(x))f_2(x)\,. \tag{3.41}$$

If $w(x), f_1(x), f_2(x)$ are all Gaussian processes (GPs), we can imagine the model accounting for arbitrarily many change-points between f_1 and f_2. Conditioned on $w(x)$, $y(x)$ is a Gaussian process with kernel

$$k(x, x') = \sigma(w(x))k_1(x, x')\sigma(w(x')) + \sigma(-w(x))k_2(x, x')\sigma(-w(x')) \,, \tag{3.42}$$

where k_1 and k_2 are the kernels of f_1 and f_2. A simple special case of the kernel in Eq. (3.42) can be obtained when $w(x) = ax^\top x + b$, a simple linear function. Saatchi et al. (2010) and Osborne (2010) contain alternative Gaussian process models for change-points.

Chapter 4

Closed Form Kernels for Pattern Discovery and Extrapolation

In chapter 3 we introduced the Gaussian process regression network (GPRN) framework, which builds expressive kernels from an adaptive mixture of base kernels. Each base kernel can be chosen to be any valid kernel. This network was used to discover input varying correlations between multiple response variables, and provides an intuitive recipe for constructing non-stationary kernels.

However, the final kernel that is induced using a GPRN, after integrating away all weights functions, does not have a closed form expression. Moreover, the Gaussian process regression network requires approximate Bayesian inference; we pursued elliptical slice sampling (ESS) and variational Bayes (VB) approximations.

In this chapter we introduce simple closed form kernels for automatic pattern discovery and extrapolation. This chapter builds on the material in Wilson and Adams (2013) and Wilson (2012), where I first introduced these kernels. These kernels are derived by modelling a spectral density – the Fourier transform of a kernel – with a Gaussian mixture. These spectral mixture (SM) kernels form a basis for all stationary kernels. Furthermore, SM kernels can be used as a drop-in replacement for popular alternative kernels, such as the Gaussian (squared exponential) kernel, with benefits in expressive power and performance, while retaining simple exact learning and inference procedures.

Gaussian process regression networks (GPRNs) and spectral mixture (SM) kernels are complementary. In this chapter, we develop powerful non-stationary kernels by using the SM kernel as the base kernel in a GPRN. The GPRN framework allows us to easily tune our inductive biases, so that the resulting model is both highly expressive, but still has the necessary preference biases to extract useful information from a wide range of datasets.

In this chapter we largely focus on univariate examples. In chapter 5 we extend the spectral mixture kernel formulation for tractability with large multidimensional patterns. We apply the resulting model, GPatt, on practically important large scale pattern extrapolation problems, including inpainting problems involving image restoration, object removal, and scene reconstruction.

Code for reproducing the experiments in section 4.3, a tutorial, and general resources for the SM kernel, including a video lecture, are available at `http://mlg.eng.cam.ac.uk/andrew/pattern`. The spectral mixture (SM) kernel has also been included in the GPML software package `http://www.gaussianprocess.org/gpml/code/matlab/doc/`.

4.1 Introduction

Machine learning is fundamentally about pattern discovery. The first machine learning models, such as the perceptron (Rosenblatt, 1962), were based on a simple model of a neuron (McCulloch and Pitts, 1943). Papers such as Rumelhart et al. (1986) inspired hope that it would be possible to develop intelligent agents with models like neural networks, which could automatically discover hidden representations in data. Indeed, machine learning aims not only to equip humans with tools to analyze data, but to fully automate the learning and decision making process.

Research on Gaussian processes (GPs) within the machine learning community developed out of neural networks research, triggered by Neal (1996), who observed that Bayesian neural networks became Gaussian processes as the number of hidden units approached infinity. Neal (1996) conjectured that “there may be simpler ways to do inference in this case”.

These simple inference techniques became the subsequent Gaussian process models for machine learning (Rasmussen and Williams, 2006) (chapter 2). These models construct a prior directly over functions, rather than parameters. Assuming Gaussian noise, one can analytically infer a posterior distribution over these functions, given data. Gaussian process models have become popular for non-linear regression and classification (Rasmussen and Williams, 2006), and often have impressive empirical performance (Rasmussen, 1996).

As discussed in detail in chapter 2, the properties of likely functions under a GP, e.g., smoothness, periodicity, etc., are specified by a positive definite *covariance kernel*[1], an operator which determines the similarity between pairs of points in the domain of the random function. The kernel controls the quality of predictions made by a GP, and the ability for the GP to extract interpretable structure from data.

Despite the importance of the kernel, and the long history of Gaussian process kernel machines in statistics,, starting with O’Hagan (1978), Gaussian processes have almost exclusively been used as smoothing interpolators with squared exponential (Gaussian) kernels.

However, Gaussian processes can help build automated intelligent systems that reason and make decisions. It has been suggested that the human ability for inductive reasoning – concept generalization with remarkably few examples – could derive from a prior combined with Bayesian inference (Steyvers et al., 2006; Tenenbaum et al., 2011; Yuille and Kersten, 2006). Bayesian nonparametric models, and Gaussian processes in particular, are an expressive way to encode prior knowledge, and can express the unbounded complexity of the real world.

Sophisticated kernels are most often achieved by composing together a few standard kernel functions (Archambeau and Bach, 2011; Durrande et al., 2011; Duvenaud et al., 2013; Gönen and Alpaydın, 2011; Rasmussen and Williams, 2006).

[1]The terms *covariance kernel*, *covariance function*, *kernel function*, and *kernel* are used interchangeably.

Tight restrictions are typically enforced on these compositions and they are hand-crafted for specialized applications. Without such restrictions, complicated compositions of kernels can lead to overfitting and unmanageable hyperparameter inference. Moreover, while some compositions (e.g., addition) have an interpretable effect, many other operations change the distribution over functions in ways that are difficult to identify. It is difficult, therefore, to construct an effective inductive bias for kernel composition that leads to automatic discovery of the appropriate statistical structure, without human intervention.

This difficulty is exacerbated by the fact that it is challenging to say anything about the covariance function of a stochastic process from a single draw if *no* assumptions are made. If we allow the covariance between any two points in the input space to arise from *any* positive definite function, with equal probability, then we gain essentially no information from a single realization. Most commonly one assumes a restriction to *stationary* kernels, meaning that covariances are invariant to translations in the input space.

In this chapter, we propose flexible classes of kernels, many of which maintain the useful inductive bias of stationarity. These kernels can be used to automatically discover patterns and extrapolate far beyond the available data. This class of kernels contains many stationary kernels, but has a simple closed form that leads to straightforward analytic inference. The simplicity of these kernels is one of their strongest qualities. In many cases, these kernels can be used as a drop in replacement for the popular squared exponential kernel, with benefits in performance and expressiveness. Moreover, by learning features in data, we not only improve predictive performance, but we can more deeply understand the structure of the problem at hand – greenhouse gases, air travel, heart physiology, brain activity, etc.

We start deriving these new kernels in section 4.2 by modelling a spectral density with a mixture of Gaussians. We focus our experiments in section 4.3 on elucidating the fundamental differences between the proposed kernels and the popular alternatives in Rasmussen and Williams (2006). In particular, we show how the proposed kernels can automatically discover patterns and extrapolate on the CO_2

dataset in Rasmussen and Williams (2006), on a synthetic dataset with strong negative covariances, on a difficult synthetic sinc pattern, and on airline passenger data. We also use our framework to reconstruct several popular standard kernels. We then introduce new expressive processes over non-stationary kernels in section 4.4, by using spectral mixture kernels with the adaptive basis functions in the Gaussian process regression network of chapter 3. For a review of Gaussian processes, see chapter 2.

4.2 Spectral Mixture (SM) Kernels

In this section we introduce a class of kernels that can discover patterns, extrapolate, and model negative covariances. This class contains a large set of stationary kernels as special cases. Roughly, a kernel measures the similarity between data points. As in Equation (2.65), the covariance kernel of a GP determines how the associated random functions will tend to vary with inputs (predictors) $x \in \mathbb{R}^P$. A *stationary kernel* is a function of $\tau = x - x'$; i.e., it is invariant to translations of the inputs.

Any stationary kernel (aka covariance function) can be expressed as an integral using Bochner's theorem (Bochner, 1959; Stein, 1999):

Theorem 4.2.1. *(Bochner) A complex-valued function k on $\mathbb{R}^P$ is the covariance function of a weakly stationary mean square continuous complex-valued random process on $\mathbb{R}^P$ if and only if it can be represented as*

$$k(\tau) = \int_{\mathbb{R}^P} e^{2\pi i s^\top \tau} \psi(\mathrm{d}s)\,, \tag{4.1}$$

where ψ is a positive finite measure.

If ψ has a density $S(s)$, then S is called the *spectral density* or *power spectrum* of k, and k and S are Fourier duals (Chatfield, 1989):

$$k(\tau) = \int S(s) e^{2\pi i s^\top \tau} ds\,, \tag{4.2}$$

$$S(s) = \int k(\tau) e^{-2\pi i s^\top \tau} d\tau\,. \tag{4.3}$$

In other words, a spectral density entirely determines the properties of a stationary kernel. Substituting the squared exponential kernel of Eq. (2.65) into Eq. (4.3), we find its spectral density is $S_{\mathrm{SE}}(s) = (2\pi\ell^2)^{P/2}\exp(-2\pi^2\ell^2 s^2)$. Likewise, the Fourier transform of a Matérn kernel is a t distribution centred at the origin. These results provide the intuition that arbitrary additive compositions of popular kernels correspond to a very small corner of the set of possible stationary kernels – equivalent to density estimation with, e.g., scale mixtures of Gaussians centred on the origin, which is not generally a model one would use for density estimation. Scale-location mixtures of Gaussians, however, can approximate any distribution to arbitrary precision with enough components (Kostantinos, 2000), and even with a small number of components are highly flexible models. That is, we can approximate *any* stationary covariance kernel to arbitrary precision, given enough mixture components in the spectral representation. These observations motivate our approach, which is to model GP covariance functions via spectral densities that are scale-location mixtures of Gaussians.

We first consider a simple case, where

$$\phi(s\,;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\{-\frac{1}{2\sigma^2}(s-\mu)^2\}, \quad \text{and} \tag{4.4}$$

$$S(s) = [\phi(s)+\phi(-s)]/2\,, \tag{4.5}$$

noting that spectral densities for real kernels must be symmetric about $s = 0$ (Hörmander, 1990). Substituting $S(s)$ into Equation (4.2) (taking the inverse Fourier transform of $S(s)$ in Eq. (4.5)) we find

$$k(\tau) = \exp\{-2\pi^2\tau^2\sigma^2\}\cos(2\pi\tau\mu)\,. \tag{4.6}$$

If $\phi(s)$ is instead a mixture of Q Gaussians on $\mathbb{R}^P$, where the q^{th} component has mean vector $\boldsymbol{\mu}_q = (\mu_q^{(1)},\dots,\mu_q^{(P)})$ and covariance matrix $V = \mathrm{diag}(v_q^{(1)},\dots,v_q^{(P)})$, w_q are mixture weights, and τ_p is the p^{th} component of the P dimensional input vector $\tau = x - x'$, then

$$k(\tau) = \sum_{q=1}^{Q} w_q \cos(2\pi\boldsymbol{\mu}_q^\top\tau)\prod_{p=1}^{P}\exp\{-2\pi^2\tau_p^2 v_q^{(p)}\}. \tag{4.7}$$

The integral in (4.2) is tractable even when the spectral density is an arbitrary Gaussian mixture, allowing us to derive[1] the exact closed form expressions in Eqs. (4.6) and (4.7), and to perform analytic inference with Gaussian processes. Moreover, this class of kernels is expressive even with a small number of components – containing many stationary kernels – but nevertheless has a simple form.

Indeed the asymptotic convergence of the kernel in Eq. (4.7) as $Q \to \infty$ to any stationary kernel is not solely why we choose to model the spectral density as a scale-location mixture of Gaussians. Many potential models can have such asymptotic convergence, but may in practice have limited flexibility with a finite number of components. With a finite Q, a scale-location mixture of Gaussians is highly flexible – unlike, for example, a finite location mixture of point masses (which would also have asymptotic convergence to any density for $Q \to \infty$.). Lázaro-Gredilla et al. (2010), for instance, consider modelling $S(s)$ as a location mixture of point masses.

Most work on spectrum estimation is specific to autoregressive models in the time domain. Such models include Bretthorst (1988); Cemgil and Godsill (2005); Qi et al. (2002), which we discuss further in section 4.4.2. As discussed in Turner (2010), the spectrum of an AR(2) process can be expressed as a Lorentzian function (in $cos(\mu)$, where μ is a spectral frequency). For AR time series models, there will be a finite number of spectral points to consider. Turner (2010) also considers placing independent inverse Gamma priors on each of these spectral points, with applications to natural sound modelling.

The kernel in Eq. (4.7) is easy to interpret, and provides drop-in replacements for kernels in Rasmussen and Williams (2006). The weights w_q specify the relative contribution of each mixture component. The inverse means $1/\mu_q$ are the component periods, and the inverse standard deviations $1/\sqrt{v_q}$ are length-scales, determining how quickly a component varies with the inputs x. The kernel in Eq. (4.7) can also be interpreted through its associated spectral density. In the experiments of the next section we use the learned spectral density to interpret

[1]Detailed derivations of Eqs. (4.6) and (4.7) are in appendix C.

the number of discovered patterns in the data and how these patterns generalize. Henceforth, we refer to the kernel in Eq. (4.7) as a spectral mixture (SM) kernel.

We can generate other new kernels from the spectral mixture (SM) kernel, using the techniques discussed in chapter 2. For example, in section 4.5, we consider an infinite scale mixture of (SM) kernels, where each component has a different bandwidth $1/\sqrt{v_q}$ parameter.

4.3 Experiments

We show how the SM kernel in Eq. (4.7) can be used to discover patterns, extrapolate, and model negative covariances. We contrast the SM kernel with popular kernels in, e.g., Rasmussen and Williams (2006) and Abrahamsen (1997), which typically only provide smooth interpolation. Although the SM kernel generally improves predictive likelihood over popular alternatives, we focus on clearly visualizing the learned kernels and spectral densities, examining patterns and predictions, and discovering structure in the data. Our objective is to elucidate the fundamental differences between the proposed SM kernel and the alternatives.

In all experiments, we model the data using a zero mean Gaussian process with additive i.i.d. Gaussian noise. As shown in section 2.3, inference in this model is exact, and the Gaussian process can be marginalised in closed form, so that the likelihood of the data can be expressed as a function of kernel hyperparameters only. We train kernel hyperparameters using nonlinear conjugate gradients to optimize the marginal likelihood $p(\boldsymbol{y}|\theta, \{x_n\}_{n=1}^N)$ of the data $\boldsymbol{y}$ given hyperparameters kernel θ, as described in section 2.3.2. We used Carl Edward Rasmussen's 2010 version of `minimize.m`.

A type of *automatic relevance determination* (MacKay, 1994; Neal, 1996) takes place during training, as a proxy for model selection. In particular, the complexity penalty in the marginal likelihood (Eq. (2.37)) can be written as a sum of eigenvalues of the covariance matrix K. The greater the weights in the SM kernel, the larger the eigenvalues of K. Therefore the weights of extraneous components – components which do not significantly improve model fit – will shrink towards zero, which helps indicate and mitigate model overspecification. We explore this property further in Figure 5.2 and section 5.4.1 of the next chapter.

Moreover, the exponential terms in the SM kernel of Eq. (4.7) have an annealing effect on the marginal likelihood, reducing multimodality in the frequency parameters, making it easier to naively optimize the marginal likelihood without converging to undesirable local optima. We discuss this annealing effect further in section 4.6. For a fully Bayesian treatment, the spectral density could alternatively be integrated out using Markov chain Monte Carlo, rather than choosing a point estimate. We consider sampling further at the end of this chapter, in section 4.4. However, we wish to emphasize that the SM kernel can be successfully used in the same way as other popular kernels, without additional inference efforts, and it is presently common practice to learn kernel hyperparameters through marginal likelihood optimization.

We compare with the popular squared exponential (SE), Matérn (MA), rational quadratic (RQ), and periodic (PE) kernels. In each comparison, we attempt to give these alternative kernels fair treatment: we initialise hyperparameters at values that give high marginal likelihoods and which are well suited to the datasets, based on features we can already see in the data. To initialise the parameters for the SM kernel, we draw length-scales and periods (inverse frequencies) from truncated Gaussian distributions with means proportional to the range of the data. We discuss initialisation further in section 4.6.

Training runtimes are on the order of minutes for all tested kernels. In these examples, comparing with multiple kernel learning (MKL) (Gönen and Alpaydın, 2011) has limited additional value. MKL is not typically intended for pattern discovery, and often uses mixtures of SE kernels. Mixtures of SE kernels correspond to scale-mixtures of Gaussian spectral densities, and do not perform well on these data, which are described by highly multimodal non-Gaussian spectral densities.

4.3.1 Extrapolating Atmospheric CO_2

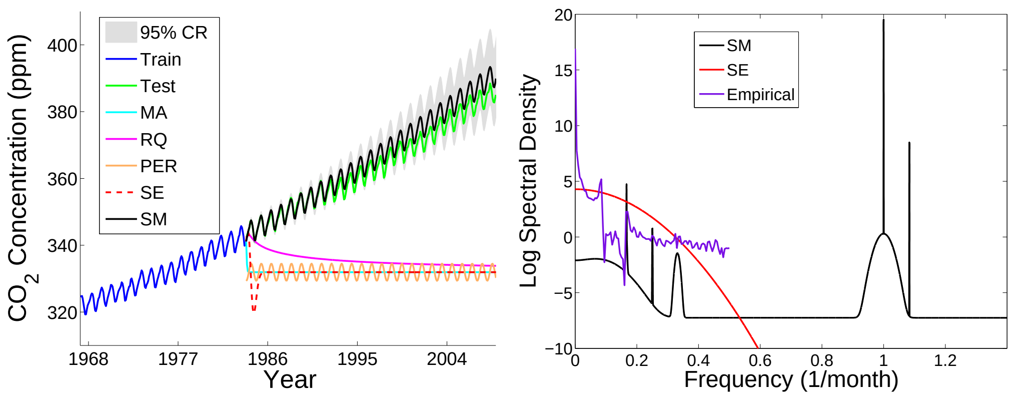
Keeling and Whorf (2004) recorded monthly average atmospheric CO_2 concentrations at the Mauna Loa Observatory, Hawaii. The data are shown in Figure 4.1. The first 200 months are used for training (in blue), and the remaining 301 months ($\approx$ 25 years) are used for testing (in green).

This dataset was used in Rasmussen and Williams (2006), and is frequently used in Gaussian process tutorials, to show how GPs are flexible statistical tools: a human can look at the data, recognize patterns, and then hard code those patterns into covariance kernels. Rasmussen and Williams (2006) identify, by looking at the blue and green curves in Figure 4.1a, a long term rising trend, seasonal variation with possible decay away from periodicity, medium term irregularities, and noise, and hard code a stationary covariance kernel to represent each of these features.

However, in this view of GP modelling, all of the interesting pattern discovery is done by the human user, and the GP is used simply as a smoothing device, with the flexibility to incorporate human inferences in the prior. Our contention is that such pattern recognition can also be performed algorithmically. To discover these patterns without encoding them *a priori* into the GP, we use the spectral mixture kernel in Eq. (4.7), with $Q = 10$. The results are shown in Figure 4.1a.

In the training region, predictions using each kernel are essentially equivalent, and entirely overlap with the training data. However, unlike the other kernels, the SM kernel (in black) is able to discover patterns in the training data and accurately extrapolate over a long range. The 95% credible region (CR) region contains the true CO_2 measurements for the duration of the measurements.

We can see the structure discovered by the SM kernel in its learned log spectral density in Figure 4.1b, in black. Of the $Q = 10$ components, only seven were used – an example of the automatic model selection we discuss further in the next chapter (e.g., see Figure 5.2). Figure 4.1b is a good example of *aliasing* in the spectral density. The sharp peak at a frequency near 1.08 corresponds to the period of 12 months, relating to pronounced yearly variations in CO_2 readings. Since 1.08 is greater than half the sampling rate of the data, 1/month, it will be aliased back to 0.08, and $1/0.08 \approx 12$ months. Since there is little width to the peak, the model is confident that this feature (yearly periodicity) should extrapolate long into the future. There is another large peak at a frequency of 1, equal to the sampling



(a) (b)

Figure 4.1: Extrapolating Mauna Loa CO_2 Concentrations. a) Forecasting CO_2. The training data are in blue, the testing data in green. Mean forecasts made using a GP with the SM kernel are in black, with 2 stdev about the mean (95% of the predictive mass, also known as the credible region (CR)) in gray shade. Predictions using GPs with Matérn (MA), rational quadratic (RQ), periodic kernels (PER), and squared exponential (SE) kernels are in cyan, magenta, orange, and dashed red, respectively. b) The log spectral densities of the learned SM and SE kernels are in black and red, respectively. The log empirical spectral density is shown in purple.

rate of the data. This peak corresponds to an aliasing of a mean function. The horizontal line (constant) at a value near -7 is noise aliasing. We can intuitively see this constant as noise aliasing by imagining a GP with an SE kernel that has a length-scale $\ell \to 0$. A draw from such a GP would be white noise, and would have a Gaussian spectral density with a variance $\to \infty$, which would appear as a constant. In these examples, aliasing will not hurt predictions, but can affect the interpretability of results. It is easy to stop aliasing, for example, by restricting the learned frequencies to be less than the Nyquist frequency (1/2 of the sampling rate of the data). For pedagogical reasons – and since it does not affect performance in these examples – we have not used such restrictions here. Finally, we see other peaks corresponding to periods of 6 months, 4 months, and 3 months. In some instances, the width of each peak can be interpreted as the model's uncertainty about the corresponding feature in the data.

In red, we show the log spectral density for the learned SE kernel, which misses many of the frequencies identified as important using the SM kernel. Like the learned SM kernel log spectral density, the log empirical spectral density, shown in purple, has pronounced maxima near the origin, and at 12 and 6 months, reflecting close correlations between nearby function values, and periodicities of 12 and 6 months. However, unlike the learned SM kernel log spectral density, the empirical log spectral density is noisy and unreliable, fluctuating wildly in value, with nothing meaningful in the data to support these fluctuations. For instance, the local optima in the empirical spectral density at 4 and 3 months are indistinguishable from noise. These fluctuations arise because the empirical spectral density is calculated without an explicit noise model, and makes few assumptions, which makes its behaviour susceptible to artifacts in the data. Stronger inductive biases are needed to reliably reconstruct the spectral density, especially for smaller datasets. Note that the empirical spectrum is only defined for $s \in [0, 0.5]$. More on aliasing and empirical spectral densities can be found in section 2.4.1.

All SE kernels have a Gaussian spectral density centred on zero. Since a mixture of Gaussians centred on the origin is a poor approximation to many density functions, combinations of SE kernels have limited expressive power. Indeed the spectral density learned by the SM kernel in Figure 4.1b is highly non-Gaussian. The test predictive performance using the SM, SE, MA, RQ, and PE kernels is given in Table 4.1, under the heading $\texttt{CO}_2$.

Eventually, predictions made using a GP with an SM kernel will revert to the prior mean. We contend that this is reasonable behaviour: the further away one gets from the data, the less we can be informed by the data, until we revert to our prior beliefs.

4.3.2 Recovering Popular Kernels

The SM class of kernels contains many stationary kernels, since mixtures of Gaussians can be used to construct a wide range of spectral densities. Even with a small number of mixture components, e.g., $Q \leq 10$, the SM kernel can closely recover popular stationary kernels catalogued in Rasmussen and Williams (2006).

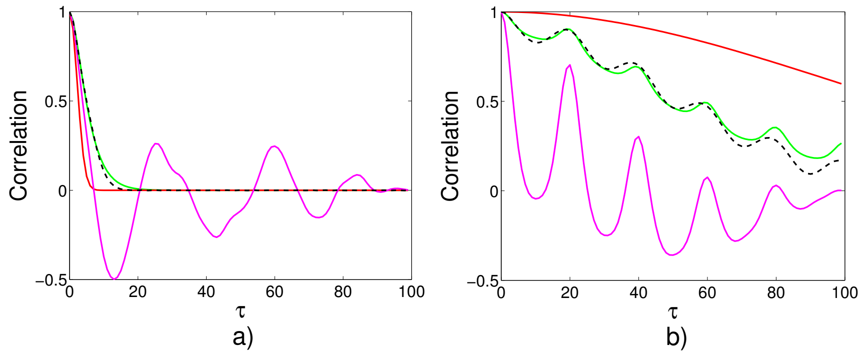


Figure 4.2: Automatically recovering popular correlation functions (normalised kernels) using the SM kernel, as a function of $\tau = x - x'$. The true correlation function underlying the data is in green, and SM, SE, and empirical correlation functions are in dashed black, red, and magenta, respectively. Data are generated from a) a Matérn kernel, and b) a sum of RQ and periodic kernels.

As an example, we start by sampling 100 points from a one-dimensional GP with a Matérn kernel with degrees of freedom $\nu = 3/2$:

$$k_{\mathrm{MA}}(\tau) = a(1 + \frac{\sqrt{3}\tau}{\ell}) \exp(-\frac{\sqrt{3}\tau}{\ell}) \,, \tag{4.8}$$

where $\ell = 5$ and $a = 4$. Sample functions from a Gaussian process with this Matérn kernel are far less smooth (only once-differentiable) than Gaussian process functions with a squared exponential kernel.

We attempt to reconstruct the kernel underlying the data by training an SM kernel with $Q = 10$. After training, only $Q = 4$ components are used. The log marginal likelihood of the data – having integrated away the Gaussian process – using the trained SM kernel is -133, compared to -138 for the Matérn kernel that generated the data. Training the SE kernel in (2.65) gives a log marginal likelihood of -140. So while the true kernel is accurately reconstructed in this case, the marginal likelihoods of the fits provide evidence that there is mild over-fitting. Such over-fitting could be mitigated by placing prior distributions over the parameters of the spectral mixture kernel, and then integrating away these parameters via MCMC, as proposed in section 4.4.1.

Figure 4.2a shows the learned SM correlation function[1], compared to the generating Matérn correlation function, the empirical autocorrelation function, and learned squared exponential correlation function. Although often used in geostatistics to guide choices of GP kernels (and parameters) (Cressie, 1993), the empirical autocorrelation function tends to be unreliable, particularly with a small amount of data (e.g., $N < 1000$), and at high lags (for $\tau > 10$). Although the empirical autocorrelation function leverages an inductive bias of stationarity, it does not have the useful bias that correlations generally decrease with distance. In Figure 4.2a, the empirical autocorrelation function is erratic and does not resemble the Matérn kernel for $\tau > 10$. Moreover, the squared exponential kernel cannot capture the heavy tails of the Matérn kernel, no matter what length-scale it has. Even though the SM kernel is infinitely differentiable, it can closely approximate processes which are finitely differentiable, because mixtures of Gaussians can closely approximate the spectral densities of these processes, even with a small number of components, as in Figure 4.2a.

Next, we reconstruct a mixture of the rational quadratic (RQ) and periodic kernels (PE) in Rasmussen and Williams (2006):

$$k_{\mathrm{RQ}}(\tau) = (1 + \frac{\tau^2}{2\,\alpha\,\ell_{RQ}^2})^{-\alpha}\,, \tag{4.9}$$

$$k_{\mathrm{PER}}(\tau) = \exp(-2\sin^2(\pi\,\tau\,\omega)/\ell_{PER}^2)\,. \tag{4.10}$$

The rational quadratic kernel in (4.9) is derived as a scale mixture of squared exponential kernels with different length-scales. The standard periodic kernel in (4.10) is derived by mapping the two dimensional variable $u(x) = (\cos(x), \sin(x))$ through the squared exponential kernel in (2.65). Derivations for both the RQ and PER kernels in Eqs. (4.9) and (4.10) are in sections 2.4.4 and 2.4.7. Rational quadratic and Matérn kernels are also discussed in Abrahamsen (1997). We sample 100 points from a Gaussian process with kernel $10k_{\mathrm{RQ}} + 4k_{\mathrm{PER}}$, with $\alpha = 2$, $\omega = 1/20$, $\ell_{RQ} = 40$, $\ell_{PER} = 1$.

[1]A *correlation function* $c(x, x')$ is a normalised covariance kernel $k(x, x')$, such that $c(x, x') = k(x, x')/\sqrt{k(x, x)k(x', x')}$ and $c(x, x) = 1$.

We reconstruct the kernel function of the sampled process using an SM kernel with $Q = 4$, with the results shown in Figure 4.2b. The heavy tails of the RQ kernel are modelled by two components with large periods (essentially aperiodic), and one short length-scale and one large length-scale. The third component has a relatively short length-scale, and a period of 20. There is not enough information in the 100 sample points to justify using more than $Q = 3$ components, and so the fourth component in the SM kernel has no effect, through the complexity penalty in the marginal likelihood. The empirical autocorrelation function somewhat captures the periodicity in the data, but significantly underestimates the correlations. The squared exponential kernel learns a long length-scale: since the SE kernel is highly misspecified with the true generating kernel, the data are explained as noise.

4.3.3 Negative Covariances

All of the stationary covariance functions in the standard machine learning Gaussian process reference Rasmussen and Williams (2006) are everywhere positive, including the periodic kernel, $k(\tau) = \exp(-2\sin^2(\pi\,\tau\,\omega)/\ell^2)$. While positive covariances are often suitable for interpolation, capturing negative covariances can be essential for extrapolating patterns: for example, linear trends have long-range negative covariances. Moreover, in the autoregressive time-series literature (appendix A), negative covariances are common. We test the ability of the SM kernel to learn negative covariances, by sampling 400 points from a simple AR(1) discrete time GP:

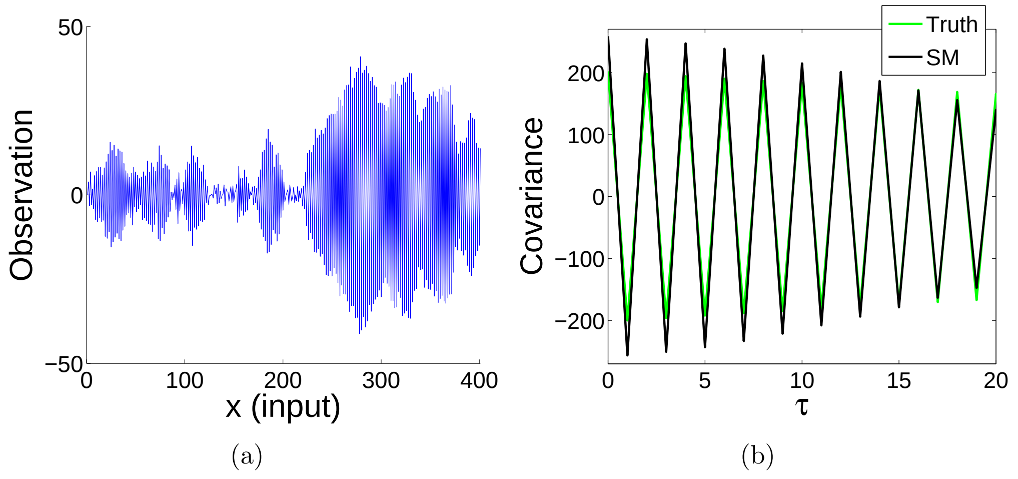
$$y(x+1) = -e^{-0.01}y(x) + \sigma\epsilon(x)\,, \tag{4.11}$$

$$\epsilon(x) \sim \mathcal{N}(0,1)\,, \tag{4.12}$$

which has kernel

$$k(x,x') = \sigma^2(-e^{-.01})^{|x-x'|}/(1-e^{-.02})\,. \tag{4.13}$$

The process in Eq. (4.11) is shown in Figure 4.3a. This process follows an oscillatory pattern, systematically switching states every $x = 1$ unit, but is not periodic



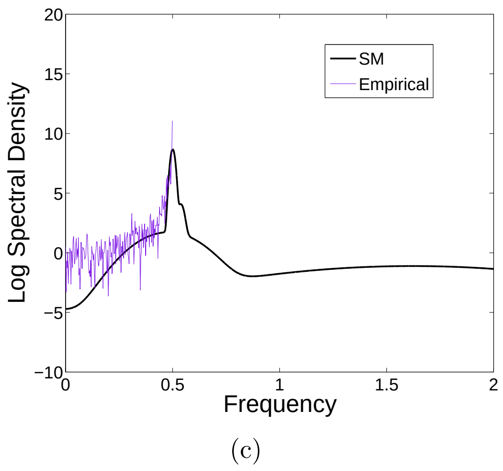


Figure 4.3: Recovering negative covariances with the SM kernel. a) Observations of a discrete time autoregressive (AR) series with negative covariances. b) The SM learned kernel is shown in black, while the true kernel of the AR series is in green, with $\tau = x - x'$. c) The log empirical and learned SM kernel spectral densities are shown in purple and black, respectively.

and has long range covariances: if we were to only view every second data point, the resulting process would vary rather slowly and smoothly.

We see in Figure 4.3b that the learned SM covariance function accurately reconstructs the true covariance function. The associated SM kernel log spectral density shown in black in Figure 4.3c has a sharp peak at a frequency of 0.5, or a period of 2. This feature represents the tendency for the process to oscillate from positive to negative every $x = 1$ unit. The log empirical spectral density, shown in purple, also has a peak at a frequency of $s = 0.5$, but has wild (and erroneous) fluctuations in value, due to a lack of reasonable inductive biases. In this example, Q was set to a value of 4 in the SM kernel, but after automatic relevance determination in training, only $Q = 3$ components were used, as can be seen by inspecting the learned SM kernel log spectral density in black.

Using a GP with a SM kernel, we forecast 20 units ahead and compare to other kernels in Table 4.1 (`NEG COV`).

4.3.4 Discovering the Sinc Pattern

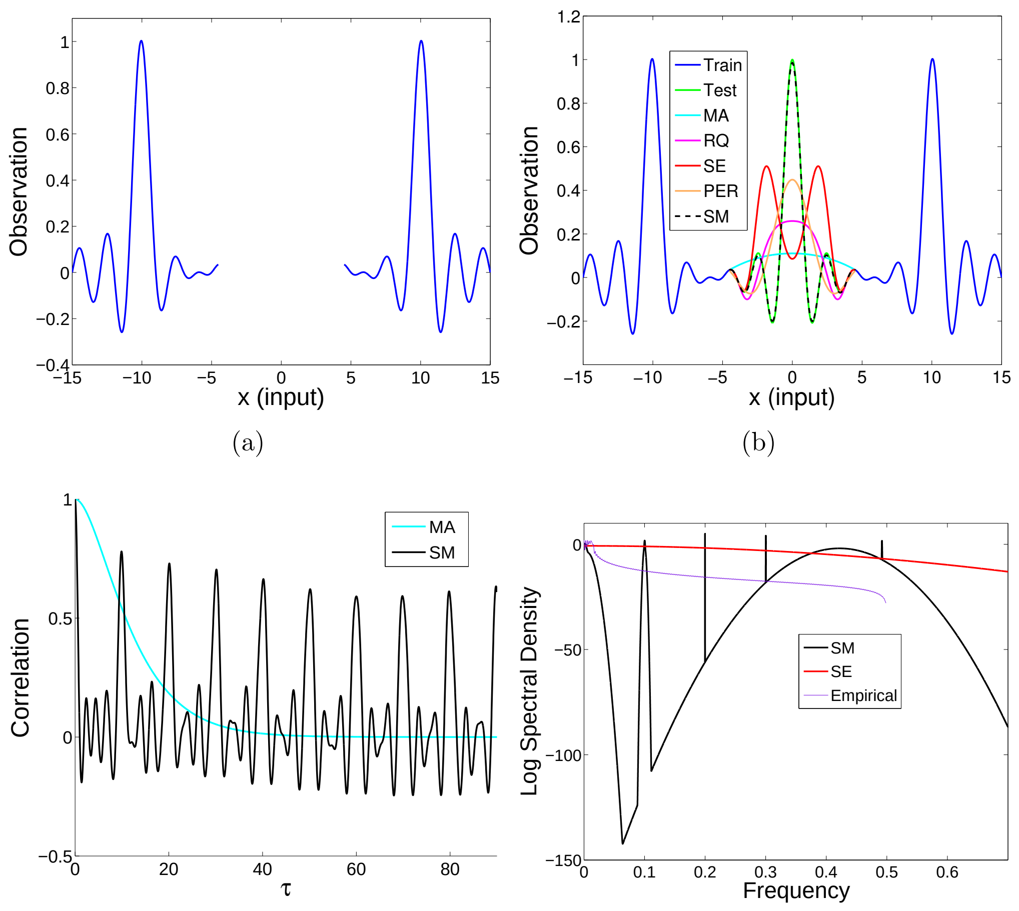
The sinc function is defined as $\text{sinc}(x) = \sin(\pi x)/(\pi x)$. We created a pattern combining three sinc functions:

$$y(x) = \text{sinc}(x + 10) + \text{sinc}(x) + \text{sinc}(x - 10)\,. \tag{4.14}$$

This is a complex oscillatory pattern. Given only the points shown in Figure 4.4a, we wish to complete the pattern for $x \in [-4.5, 4.5]$. Unlike the CO_2 example in section 4.3.1, it is perhaps even difficult for a human to extrapolate the missing pattern from the training data. It is an interesting exercise to focus on this figure, identify features, and fill in the missing part.

Notice that there is complete symmetry about the origin $x = 0$, peaks at $x = -10$ and $x = 10$, and destructive interference on each side of the peaks facing the origin. We therefore might expect a peak at $x = 0$ and a symmetric pattern around $x = 0$.

As shown in Figure 4.4b, the SM kernel with $Q = 10$ reconstructs the pattern in the region $x \in [-4.5, 4.5]$ almost perfectly from the 700 training points in blue. Moreover, using the SM kernel, 95% of the posterior predictive mass entirely contains the true pattern in $x \in [-4.5, 4.5]$. GPs using Matérn, SE, RQ, and periodic kernels are able to predict reasonably within $x = 0.5$ units of the training data, but entirely miss the pattern in $x \in [-4.5, 4.5]$.



(a) (b)

(c) (d)

Figure 4.4: Discovering a sinc pattern with the SM kernel. a) The training data are shown in blue. The goal is to fill in the missing region $x \in [-4.5, 4.5]$. b) The training data are in blue, the testing data in green. The mean of the predictive distribution using the SM kernel is shown in dashed black. The mean of the predictive distributions using the Matérn (MA), rational quadratic (RQ), squared exponential (SE), and periodic kernels (PER), are in cyan, magenta, red, and orange. c) The learned SM correlation function (normalised kernel) is shown in black, and the learned Matérn correlation function is in cyan, with $\tau = x - x'$. d) The learned log spectral densities of the SM and SE kernels are respectively in black and red, and the log empirical spectral density is in purple.

Figure 4.4c shows the learned SM correlation function (normalised kernel). For $\tau \in [0, 10]$ there is a local pattern, roughly representing the behaviour of a single sinc function. For $\tau > 10$ there is a repetitive pattern representing a new sinc function every 10 units – an extrapolation a human might make. With a sinc functions centred at $x = -10, 0, 10$, we might expect more sinc patterns every 10 units. The learned Matérn correlation function is shown in red in Figure 4.4c. Unable to discover complex patterns in the data, the learned Matérn kernel simply assigns high correlation to nearby points.

In fact, the SM kernel reconstruction of the missing sinc function is so accurate that it might arouse suspicion of over-fitting. Surely the reconstruction in dashed black is only one of many functions that could describe the missing region. These alternate solutions could be represented in the reconstruction by placing a prior on SM kernel hyperparameters, and then sampling from the posterior over kernels induced by this prior, as described in section 4.4. On the other hand, the sinc function is an extremely likely candidate under this model, especially with the inductive bias of stationarity. Therefore sampling might not have a profound effect. If we had to choose the single most likely candidate for the missing region, then it is reasonable to perform the extrapolation in dashed black. If we wished to minimize the expected squared loss of our reconstruction, for example, then we would not so exactly reconstruct a sinc function in $[-5, 5]$. Also note in Table 4.1 that the *predictive* test likelihood (which penalizes overconfidence) on this sinc example, under a GP with an SM kernel, is significantly higher than for the alternatives drawn in Figure 4.4b, which suggests that over-fitting may be such a major problem in this example. Moreover, an informal lecture survey suggested that the dashed black is close to what a person would choose if extrapolating the region in $[-5, 5]$. What would you draw in the missing region of Figure 4.4a?

Figure 4.4d shows the (highly non-Gaussian) log spectral density of the SM kernel in black, with peaks at $0.003, 0.1, 0.2, 0.3, 0.415, 0.424, 0.492$. In this case, only 7 of the $Q = 10$ components are used. The peak at 0.1 represents a period of 10: every 10 units, a sinc function is repeated. The variance of this peak is small, meaning the method will extrapolate this structure over long distances. By contrast, the log squared exponential spectral density in red simply has a broad peak, centred at the origin. The log empirical spectral density in purple is unable to capture the same features as the spectral mixture kernel. Here the empirical spectral density is calculated assuming the gap in the training data does not exist; however, the empirical spectral density corresponding to the joined training and test sets behaves similarly.

The predictive performance for recovering the missing 300 test points (in green) is given in Table 4.1 (`SINC`).

4.3.5 Airline Passenger Data

Figure 4.5a shows airline passenger numbers, recorded monthly, from 1949 to 1961 (Hyndman, 2005). Based on only the first 96 monthly measurements, in blue, we wish to forecast airline passenger numbers for the next 48 months (4 years); the corresponding 48 test measurements are in green.

There are several features apparent in these data: short seasonal variations, a long term rising trend, and an absence of white noise artifacts. Many popular kernels are forced to make one of two choices: 1) Model the short term variations and ignore the long term trend, at the expense of extrapolation. 2) Model the long term trend and treat the shorter variations as noise, at the expense of interpolation.

As seen in Figure 4.5a, the Matérn kernel is more inclined to model the short term trends than the smoother SE or RQ kernels, resulting in sensible interpolation (predicting almost identical values to the training data in the training region), but poor extrapolation – moving quickly to the prior mean, having learned no structure to generalise beyond the data. The SE kernel interpolates somewhat sensibly, but appears to underestimate the magnitudes of peaks and troughs, and treats repetitive patterns in the data as noise. Extrapolation using the SE kernel is poor. The RQ kernel, which is a scale mixture of SE kernels, is more able to manage different length-scales in the data, and generalizes the long term trend better than the SE kernel, but interpolates poorly. The sparse spectrum Gaussian process (SSGP) (Lázaro-Gredilla et al., 2010), shown in dark green, also performs poorly on this dataset (the fit shown is the best from a variety of hyperparameter initialisations). The SSGP models the spectral density as a sum of point masses, and is a finite basis function model with as many basis functions as point masses. Although in principle the SSGP can model any spectral density (and thus any stationary kernel) with an infinite number of point masses, SSGP has very limited expressive power with a finite number of point masses, as a finite number of point masses is limited in its ability to perform density estimation, and will be prone to over-fitting. More comparisons to SSGP, which provides fast and flexible kernel learning, are shown in chapter 5.

By contrast, the SM kernel interpolates nicely (overlapping with the data in the training region), and is able to extrapolate complex patterns far beyond the data, capturing the true airline passenger numbers for years after the data ends, within a small band containing 95% of the predictive probability mass.

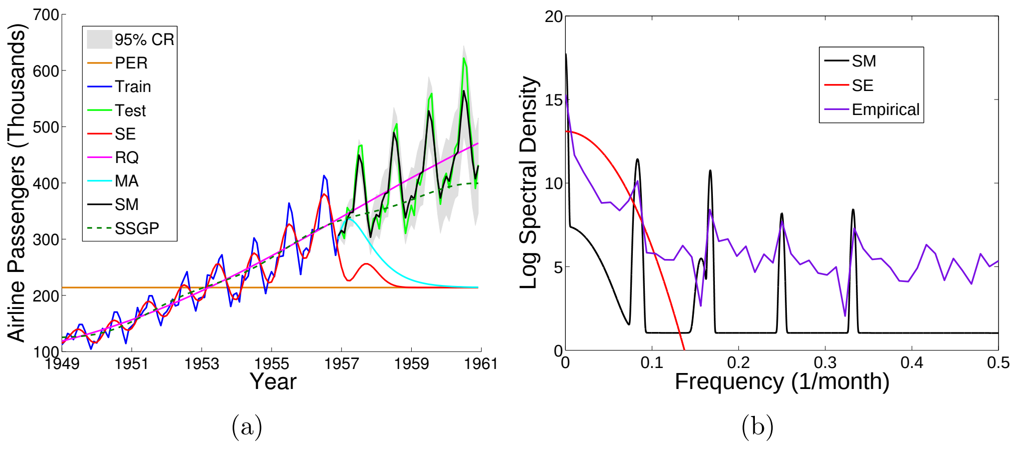


Figure 4.5: Predicting airline passenger numbers using the SM kernel. a) The training and testing data are in blue and green, respectively. The mean of the predictive distribution using the SM kernel is shown in black, with 2 standard deviations about the mean (95% of the predictive probability mass, aka the credible region (CR)) shown in gray shade. The mean of the predictive distribution using the SM kernel is shown in black. The mean of the predictive distributions using GPs with periodic (PER), squared exponential (SE), rational quadratic (RQ), and Matérn (MA) kernels are in orange, red, magenta, and cyan, respectively. Predictions using SSGP are shown in dashed dark green. More comparisons involving SSGP are in chapter 5. In the training region, the SM and Matérn kernels are not shown, since their predictions essentially overlap with the training data. b) The log spectral densities of the SM and squared exponential kernels are in black and red, respectively. The empirical log spectral density is shown in purple.

Of the $Q = 10$ initial components specified for the SM kernel, 7 were used after training. The learned log spectral density in Figure 4.5b shows a large sharp low frequency peak (at about 0.00148). This peak corresponds to the rising trend, which generalises well beyond the data (small variance peak), is smooth (low frequency), and is important for describing the data (large relative weighting). The next largest peak corresponds to the yearly trend of 12 months, which again generalises, but not to the extent of the smooth rising trend, since the variance of this peak is larger than for the peak near the origin. The higher frequency peak at $x = 0.34$ (period of 3 months) corresponds to the beginnings of new seasons, which can explain the effect of seasonal holidays on air traffic. The log empirical spectral density (in purple) resembles the learned SM spectral density, with pronounced optima in the same locations, which is a reassuring verification that the SM kernel has discovered the relevant features in the data. However, the empirical spectral density behaves erratically, with many peaks and troughs which do not correspond to any interpretable features in the data.

A more detailed study of these features and their properties – frequencies, variances, etc. – could isolate less obvious features affecting airline passenger numbers.

Table 4.1 (`AIRLINE`) shows predictive performance for forecasting 48 months of airline passenger numbers.

4.4 Processes over Kernels for Pattern Discovery

In this section, we briefly introduce processes over kernels, including a process over all non-stationary kernels, using the Gaussian process regression network (chapter 3) as a starting point. We incorporate these processes into hierarchical Gaussian process models, derive inference procedures for these models, and provide a demonstration on the airline dataset, which illustrates several different types of non-stationarity. In general, non-stationary processes over kernels is a promising future direction for the body of work presented in this chapter.

With a process over kernels, one can naturally represent uncertainty in the kernel function, and precisely control the inductive biases in a Gaussian process based model. Placing a prior distribution over the hyperparameters of a kernel induces a

Table 4.1: We compare the test performance of the proposed spectral mixture (SM) kernel with squared exponential (SE), Matérn (MA), rational quadratic (RQ), and periodic (PE) kernels. The SM kernel consistently has the lowest mean squared error (MSE) and highest log likelihood ($\mathcal{L}$).

	SM	SE	MA	RQ	PE
`CO2`					
MSE	**9.5**	1200	1200	980	1200
$\mathcal{L}$	**170**	−320	−240	−100	−1800
`NEG COV`					
MSE	**62**	210	210	210	210
$\mathcal{L}$	**−25**	−70	−70	−70	−70
`SINC`					
MSE	**0.000045**	0.16	0.10	0.11	0.05
$\mathcal{L}$	**3900**	2000	1600	2000	600
`AIRLINE`					
MSE	**460**	43000	37000	4200	46000
$\mathcal{L}$	**−190**	−260	−240	−280	−370

process prior over kernels. It is natural to think of this process over kernels in the same way we think of processes over functions in function space regression (chapter 2). We can sample from a prior over kernel hyperparameters, and then condition on these samples, to draw sample kernels (aka covariance functions). Given data, we can sample from a posterior over kernel hyperparameters, and therefore we can also draw samples from the induced posterior distribution over kernels.

4.4.1 Processes over SM Kernels

We induce a process prior over stationary kernels, $p(k_{\mathrm{S}})$, by placing prior distributions on the parameters $\theta = \{a_q, \sigma_q, \mu_q\}_{q=1}^{Q}$ of the spectral mixture (SM) kernel in Equation (4.7), such that $k_{\mathrm{S}}|\theta = k_{\mathrm{SM}}$ (we have switched notation from w_q to a_q for the mixture weights, for clarity in the proceeding discussion), and performing Bayesian inference over SM kernel parameters. For demonstration purposes, we will initially use the following prior distributions:

$$p(a_q) = \mathrm{Gamma}(a_q|2,4)\,, \qquad p(1/\sigma_q) = \mathrm{Gamma}(a_q|10,10)\,, \tag{4.15}$$

where we parametrize the Gamma density as

$$\mathrm{Gamma}(x|\alpha,\beta) = \frac{x^{\alpha-1}e^{-x/\alpha}}{\beta^{\alpha}\Gamma(\beta)}\,, \qquad x \geq 0\,, \quad \alpha,\beta > 0 \tag{4.16}$$

$$\mathbb{E}[x] = \alpha\beta\,, \tag{4.17}$$

$$\mathrm{V}[x] = \alpha\beta^2\,. \tag{4.18}$$

We place an isotropic (spherical) Gaussian prior on the frequency parameters μ_q with a mean vector ranging uniformly from 4 to 200, and a covariance matrix of $20^2 I$. For demonstration purposes we set the number of mixture components in the SM kernel to $Q = 10$.

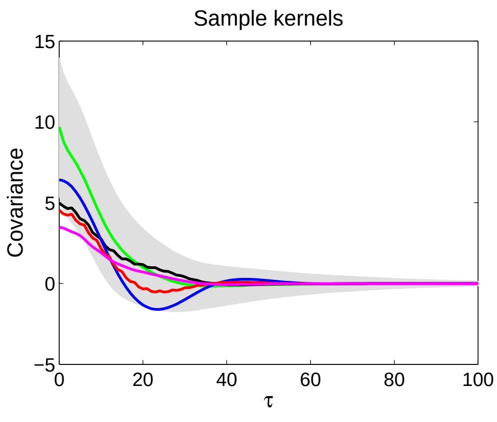
Samples of kernel functions from $p(k_{\mathrm{S}})$ are shown in Figure 4.6a. One can incorporate this process prior over kernels into a hierarchical Gaussian process model as follows:

$$k_{\mathrm{S}} \sim p(k_{\mathrm{S}}) \tag{4.19}$$

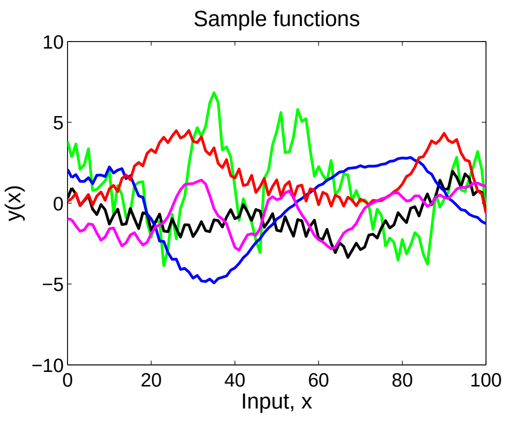
$$f(x)|k_{\mathrm{S}} \sim \mathcal{GP}(0, k_{\mathrm{S}}) \tag{4.20}$$

$$y(x)|f(x) = f(x) + \epsilon(x)\,, \tag{4.21}$$

where $\epsilon(x) \sim \mathcal{N}(0, v_\epsilon)$ is i.i.d. Gaussian white noise. Samples from $y(x)$ are shown in Figure 4.6b. We show in section 4.4.1 that the unconditional process for $y(x)$ is typically non-Gaussian.



(a)



(b)

Figure 4.6: Processes over kernels. a) Sample kernels drawn from k_{S}, a process over kernels in Equation. (4.19) using the kernel hyperpriors in Eq. (4.15). $\tau = x - x' \in \mathbb{R}^1$. The mean kernel is shown in green and 2 standard deviations about the mean kernel is shown in gray shade (both determined from 10000 samples). Sample kernels are shown in blue, black, red, and purple. b) Sample functions drawn using the kernels in panel a): the colour of the function correponds to the kernel used in Figure a). The green function, for example, is sampled using the green (mean) kernel in Figure a).

Conditioned on the kernel hyperparameters θ, we assume the data $\mathcal{D}$ are sampled from a Gaussian process $y(x)$ with kernel $k_{\mathrm{SM}} = k_{\mathrm{S}}|\theta + \delta_{ij}v_\epsilon$. The unconditional process posterior $y(x_*)$, evaluated at test input(s) $x = x_*$, is given by

$$p(y(x_*)|\mathcal{D}) = \int p(y(x_*)|\theta, \mathcal{D})p(\theta|\mathcal{D})\,, \tag{4.22}$$

$$= \lim_{J\to\infty} \frac{1}{J}\sum_{i=1}^{J} p(y(x_*)|\mathcal{D}, \theta_i)\,, \qquad \theta_i \sim p(\theta|\mathcal{D})\,, \tag{4.23}$$

where $p(y(x_*)|\mathcal{D}, \theta_i)$ is the standard predictive distribution for a Gaussian process, given in Eq. (2.27), with $N \times N$ covariance matrix $K_{ij} = k_{\mathrm{SM}}(x_i, x_j|\theta_i)$. Since the sum of Eq. (4.23) is an infinite scale-location mixture of Gaussians, the distribution $p(y(x_*)|\mathcal{D})$, and hence the process $y(x)|\mathcal{D}$, can be highly non-Gaussian. Equation (4.23) holds if we do not condition on the data $\mathcal{D}$, and so the prior process $p(y(x))$ can also be highly non-Gaussian.

Similarly, a process prior over stationary kernels $p(k_{\mathrm{S}})$ is defined by the spectral mixture (SM) kernel k_{SM} in Eq. (4.7) together with the hyperprior $p(\theta)$ in Eq. (4.15). The posterior process $p(k_{\mathrm{S}}|\mathcal{D})$ over stationary kernels is given by

$$p(k_{\mathrm{S}}|\mathcal{D}) = \int p(k_{\mathrm{S}}|\theta)p(\theta|\mathcal{D})d\theta \,, \tag{4.24}$$

$$= \lim_{J\to\infty} \frac{1}{J}\sum_{i=1}^{J} p(k_{\mathrm{S}}|\theta_i)\,, \qquad \theta_i \sim p(\theta|\mathcal{D})\,. \tag{4.25}$$

Since each component of the mixtures in (4.23) and (4.25) has equal weighting $1/J$, if we draw a sample from the posterior distribution over hyperparameters $p(\theta|\mathcal{D})$, we can condition on that sample to draw a sample from $p(y(x_*)|\mathcal{D})$ or $p(k_{\mathrm{S}}|\mathcal{D})$.

One can sample from a posterior over hyperparameters,

$$p(\theta|\mathcal{D}) \propto p(\mathcal{D}|\theta)p(\theta)\,, \tag{4.26}$$

using, for example, slice sampling (Neal, 2003). $p(\mathcal{D}|\theta)$ is the marginal likelihood of the Gaussian process $y(x)|\theta$ in Eq. (4.21), with $N \times N$ covariance matrix $K_{ij} = k_{\mathrm{SM}}(x_i, x_j|\theta)$, and $p(\theta)$ is the prior over SM kernel hyperparameters.

4.4.2 Processes over Non-Stationary Kernels

To construct a non-stationary kernel, k_{NS}, we can imagine the process given by Equation (4.21) is modulated by a function $w(x)$:

$$y(x) = w(x)(f(x) + \epsilon(x)) + \gamma(x)\,, \tag{4.27}$$

where $\gamma(x) \sim \mathcal{N}(0, v_\gamma)$.[1] The induced non-stationary (NS) kernel for the Gaussian process $y(x)$ given $w(x)$ and hyperparameters θ are

$$k_{\text{NS}}(x,x')|w(x),\theta = w(x)k_{\text{SM}}(x,x')w(x') + (w(x)w(x')v_\epsilon + v_\gamma)\delta_{xx'}\,, \tag{4.28}$$

where $\delta_{xx'}$ is the Kronecker delta function. We let

$$w(x) = \log[1+\exp(z(x))]\,, \tag{4.29}$$

$$z(x) \sim \mathcal{GP}(0, k_{\text{SE}})\,, \tag{4.30}$$

where k_{SE} is the squared exponential kernel in (2.65). Increasing the *length-scale* hyperparameter of k_{SE} concentrates the support of $w(x)$ around constant functions, and thus concentrates the support of k_{NS} on stationary kernels. Therefore k_{NS} can leverage the inductive bias of stationarity to extract useful information from the data, yet has the ability to respond to a wide range of non-stationarity through $w(x)$, which has support for any positive continuous function[2].

We can further extend the model of $y(x)$ as

$$y(x)|\{w_q(x), f_q(x)\}_{q=1}^{Q} = \sum_{q=1}^{Q} w_q(x) f_q(x)\,, \tag{4.31}$$

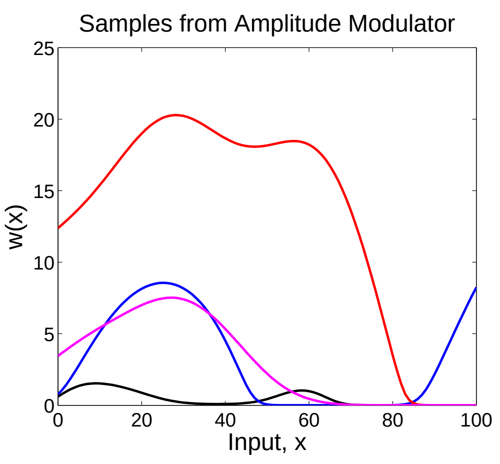
where each modulating $w_q(x)$ is specified by Eqs. (4.29)-(4.30), and each $f_q(x)$ is a GP with kernel $k(\tau) = \exp\{-2\pi^2\tau^2\sigma_q^2\}\cos(2\pi\tau\mu_q)$. The kernel $g_{\text{NS}}(x,x')$ for $y(x)$ conditioned on $\{w_q(x)\}_{q=1}^{Q}$ is

$$g_{\text{NS}}(x,x') = \sum_{q=1}^{Q} w(x)\exp\{-2\pi^2\tau^2\sigma_q^2\}\cos(2\pi\tau\mu_q)w(x')\,. \tag{4.32}$$

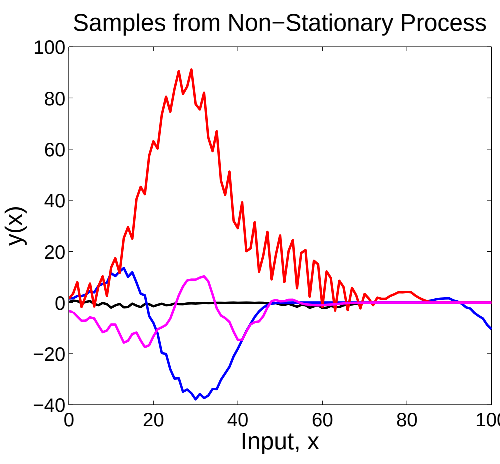
For any pair of inputs x, x', the kernel in Eq. (4.32) is a process over spectral mixture kernels, as in Eq. (4.7), which has support for any stationary kernel, given sufficiently many components Q. Therefore the kernel $g_{\text{NS}}(x,x')$ has support for

[1]Natural sounds, for example, can often be viewed as stationary patterns modulated by envelope functions (Turner, 2010).

[2]Strictly speaking, $y(x)$ in Eq. (4.27) is a stationary process after integrating away $w(x)$, since $w(x)$ is a Gaussian process with a stationary kernel. However $y(x)$ has support for highly nonstationary data, since $w(x)$ has large support for non-constant functions.



(a)



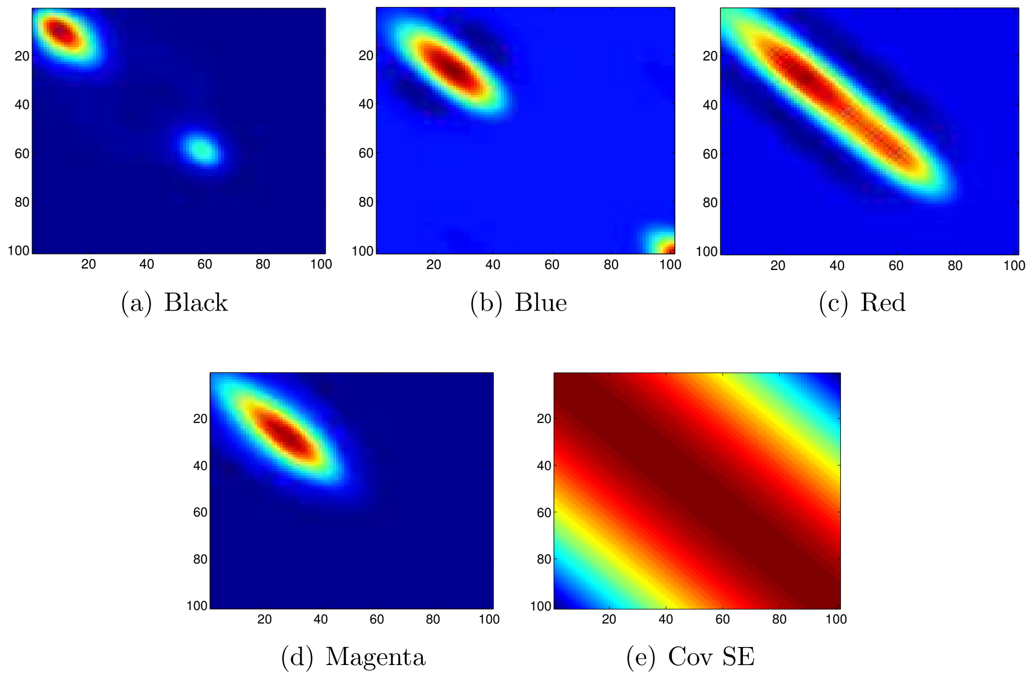
(b)

Figure 4.7: Processes over non-stationary kernels. a) Sample amplitude modulators $w(x)$ drawn using Equation (4.29). b) Sample functions from the non-stationary process defined by Equation (4.27). These sample functions are the same sample functions as in Figure 4.6, but modulated by the sample functions in $w(x)$, following the same colour scheme as in Figure 4.6.

any non-stationary kernel, given sufficiently large Q, since all non-stationary kernels are locally stationary. One can concentrate the support of $g_{\text{NS}}(x, x')$ around stationary kernels, by concentrating the support of $\{w_q\}_{q=1}^{Q}$ around constant functions, so that the process over $g_{\text{NS}}(x, x')$ has a sensible inductive bias, but is still capable of responding to any type of non-stationarity in the data. Notice that Eq. (4.31) is an example of a Gaussian process regression network (chapter 3)! Indeed the GPRN induces a process over kernels, and if the node functions use spectral mixture kernels, as in Eq. (4.32), then the induced process over kernels is extremely flexible.

The non-stationary model in Equation (4.27) assumes the data $\mathcal{D}$ are sampled from a Gaussian process $y(x)|w(x), \theta$ with kernel $k_{\text{NS}}(x, x')|w(x), \theta$, conditioned on the spectral mixture hyperparameters θ, and $w(x)$, an amplitude modulating function. $w(x)$ is deterministically related to the Gaussian process $z(x)$ through $w(x) = \log[1 + \exp(z(x))]$, as in Eq. (4.29).

The process posterior $y(x_*)|\mathcal{D}$, evaluated at test input(s) $x = x_*$, and conditioned


(a) Black (b) Blue (c) Red

(d) Magenta (e) Cov SE

Figure 4.8: Non-stationary covariance matrices. All elements of the $N \times N$ covariance matrices of the random functions shown in Figure 4.7, labelled with the colour of each corresponding function (here $N = 101$ and $\tau = 0, 1, \ldots, 100$). For comparison, panel e) shows an $N \times N$ stationary SE covariance matrix, with a length-scale of 100, the mean length-scale of the Gamma distribution used as a prior over length-scales in Equation (4.15).

only on data $\mathcal{D}$ (and x_*), is given by

$$p(y(x_*)|\mathcal{D}) = \int p(y(x_*)|z(x), \theta, \mathcal{D})p(z(x), \theta|\mathcal{D})\,, \tag{4.33}$$

$$= \lim_{J \to \infty} \frac{1}{J} \sum_{i=1}^{J} p(y(x_*)|z(x), \mathcal{D}, \theta_i)\,, \qquad \theta_i \sim p(z(x), \theta|\mathcal{D})\,, \tag{4.34}$$

where $p(y(x_*)|z(x), \mathcal{D}, \theta_i)$ is the predictive distribution for a Gaussian process, as in Eq. (2.27), with covariance kernel $k_{\text{NS}}(x, x')|w(x), \theta$ given in Eq. (4.28). Since $p(y(x_*)|\mathcal{D})$ is an infinite scale-location mixture of Gaussians, as in Eq. (4.34), the posterior process $y(x)|\mathcal{D}$ can be highly non-Gaussian. If we do not condition on data $\mathcal{D}$ in (4.34), we see the prior process $p(y(x))$ is also an infinite scale-location mixture of Gaussians.

The posterior process over non-stationary kernels $p(k_{\mathrm{NS}}|\mathcal{D})$, with process prior $p(k_{\mathrm{NS}})$ defined by Eq. (4.28), is given by

$$p(k_{\mathrm{NS}}|\mathcal{D}) = \int p(k_{\mathrm{NS}}|\theta, z(x))p(\theta, z(x)|\mathcal{D})d\theta dz(x)\,, \tag{4.35}$$

$$= \lim_{J\to\infty} \frac{1}{J}\sum_{i=1}^{J} p(k_{\mathrm{NS}}|\theta_i, z_i(x))\,, \qquad \theta_i, z_i(x) \sim p(\theta, z(x)|\mathcal{D})\,. \tag{4.36}$$

Therefore if we can sample from the joint distribution $p(\theta, z(x)|\mathcal{D})$, then we can sample from the posteriors $p(y(x_*)|\mathcal{D})$ and $p(k_{\mathrm{NS}}|\mathcal{D})$ in Eqs. (4.34) and (4.36). To sample from $p(\theta, z(x)|\mathcal{D})$ we follow a Gibbs sampling procedure (Geman and Geman, 1984), where we initialize θ and $\boldsymbol{z} = \{z(x_i)\}_{i=1}^{N}$, and, until convergence, alternately sample from the conditional posteriors

$$p(\theta|\boldsymbol{z}, \mathcal{D}) \propto p(\mathcal{D}|\theta, \boldsymbol{z})p(\theta)\,, \tag{4.37}$$

$$p(\boldsymbol{z}|\theta, \mathcal{D}) \propto p(\mathcal{D}|\theta, \boldsymbol{z})p(\boldsymbol{z})\,. \tag{4.38}$$

The Gaussian process predictive distribution $p(z(x)|\boldsymbol{z})$, for any input x, is given by Equation (2.27), where $z(x)$ has the squared exponential kernel in Eq. (2.65). $p(\mathcal{D}|\theta, \boldsymbol{z})$ is equivalent to the marginal likelihood of a Gaussian process $y(x)|w(x)$, with $N \times N$ covariance matrix $C = DKD$, where D is a diagonal matrix with $D_{ii} = w(x_i)$, and $K_{ij} = k_{\mathrm{SM}}(x_i, x_j)$ as defined in Equation (4.7). To sample from the posterior $p(\boldsymbol{z}|\theta, \mathcal{D})$ in Eq. (4.38), we use elliptical slice sampling (Murray et al., 2010), which was especially designed to sample from posteriors with correlated Gaussian priors, such as $p(\boldsymbol{z})$. We sample from the posterior over $p(\theta|\boldsymbol{z}, \mathcal{D})$ in Eq. (4.37) using slice sampling (Neal, 2003). One can also learn the hyperparameters θ_w of $w(x)$ by adding an additional step to the Gibbs procedure in Eqs. (4.37)-(4.38), to sample from $p(\theta_w|z(x)) \propto p(z(x)|\theta_w)p(\theta_w)$.

Inference for the non-stationary model of Eq. (4.31) proceeds similarly. As an alternative, the variational Bayes inference techniques used in chapter 3 would apply with modification to the model in Eq. (4.31). One would need to account for the non-linearity introduced by the link function $w(z) = \log(1 + e^z)$.

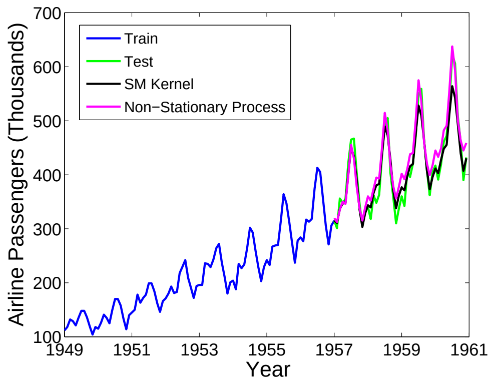


Figure 4.9: Extrapolating airline passenger numbers using a non-stationary process. This airline passenger data has several types of non-stationarity. The linear-trend is an example of one obvious non-stationarity (and we would likely expect the trend to roughly asymptote), and the increasing magnitude of the peaks and troughs is another non-stationarity. The proposed non-stationary process of Eq. (4.27), in magenta, is more able to capture the latter non-stationarity (particularly the increasing peaks) than a GP using a spectral mixture kernel (black). Training data are in blue, and testing data in green. Alex Matthews provided assistance in creating this figure.

4.4.3 Application to Airline Passenger Data

We apply the non-stationary process over kernels in Equation (4.27), with the inference described in section 4.4.2, to the airline dataset, with results shown in Figure 4.9. In this example, we use log uniform priors on all spectral mixture kernel hyperparameters. Notice that the airline dataset contains several types of non-stationary behaviour. There is both a rising trend, and the magnitudes of the peaks and troughs are increasing with the inputs. The latter type of non-stationarity is especially well handled by the amplitude modulating function $w(x)$. We can see the peaks in the airline dataset are now properly extrapolated using the proposed process over non-stationary kernels.[1]

[1]For the SM kernel results in Figure 4.9, hyperparameters were trained with marginal likelihood optimization. A similar result is achieved with hyperparameter sampling.

4.4.4 Fast Toeplitz Inference and Audio Modelling

In the future it would be interesting to apply the proposed non-stationary processes over kernels to modelling audio data, which can often be fruitfully viewed as stationary carrier patterns modulated by envelope functions (Turner, 2010).

Typically audio samples contain more than 10^5 training instances. Moreover, audio data is typically on a regularly sampled grid, with one dimensional inputs representing time. In this case, Toeplitz structure can be used to scale the proposed processes over non-stationary kernels to large datasets, without any loss in predictive accuracy.

We propose to perform elliptical slice sampling (Murray et al., 2010) inference on the amplitude modulator(s) $w(x)$. Elliptical slice sampling requires that one can sample from the prior over $w(x)$ and evaluate the likelihood of the model conditioned on $w(x)$. Likelihood evaluations, conditioned on $w(x)$ and $f(x)$, requires $\mathcal{O}(N)$ storage and operations, where N is the number of training datapoints. Sampling from the prior over $w(x)$ at the N training locations of interest requires that we sample from $\mathcal{N}(0, K_w)$, where the $N \times N$ matrix $[K_w]_{ij} = k_{\text{SE}}(x_i, x_j)$. Naively, sampling $w(x)$ involves: i) sampling white noise $\nu \sim \mathcal{N}(0, I)$ from built-in routines, an $\mathcal{O}(N)$ operation, ii) forming the product $L\nu$, where $\nu \sim \mathcal{N}(0, I)$ and L is the lower cholesky decomposition of K_w, such that $LL^\top = K_w$. This Cholesky decomposition costs $\mathcal{O}(N^2)$ storage and $\mathcal{O}(N^3)$ operations.

We can sample from $w(x)$ more efficiently if we exploit the Toeplitz structure in K_w.[1] First we embed K_w into a circulant matrix C_w, following Eq. (2.78). Letting the DFT be the discrete Fourier transform of a vector, we compute $\tilde{\nu} = \text{DFT}[\nu]$ and $\tilde{c} = \text{DFT}[c]$, where c is the first column of C_w. Now, a Gaussian variable with a circulant covariance matrix in the time domain is a Gaussian variable with a diagonal covariance matrix in the frequency domain. E.g.,

$$C\nu = \text{DFT}^{-1}[\text{diag}(\tilde{c})\text{DFT}[\nu]] \,. \tag{4.39}$$

[1]section 2.6.3.1 introduces the basics of exploiting Toeplitz structure to speed up computations.

And therefore a sample from $\mathcal{N}(0, C_w)$ can be obtained as

$$C^{1/2}\nu = \text{DFT}^{-1}[\tilde{c}^{1/2}\tilde{\nu}] \,. \tag{4.40}$$

Recalling that the Gaussian distribution has a marginalization property – examination of any set of Gaussian random variables does not change the distribution of any subset – we can discard the additional $N+1$ points added to the original K_w to form a circulant embedding, and obtain a draw from $\mathcal{N}(0, K_w)$ with a Toeplitz covariance matrix. This sampling procedure only requires $\mathcal{O}(N)$ memory and $\mathcal{O}(N \log N)$ computations: we only need to store the first column of an $N \times N$ matrix and an $N \times 1$ vector, and the discrete Fourier transform (DFT) of an $N \times 1$ vector can be computed in $\mathcal{O}(N \log N)$ using fast Fourier transforms (FFTs)!

We can also exploit Toeplitz structure to efficiently learn the spectral mixture hyperparameters θ in the non-stationary model. Sampling from $p(\theta|\boldsymbol{z}, \mathcal{D})$ involves inverting the covariance matrix $A = DKD$, where D is a diagonal matrix with $D_{ii} = w(x_i)$, and $K_{ij}|\theta = k_{\text{SM}}(x_i, x_j)$. Since K is Toeplitz, this inversion can be computed in $\mathcal{O}(N)$ memory and $\mathcal{O}(N \log N)$ computations (section 2.6.3.1). Note that even if $A = DKD + \sigma^2 I$, so as to incorporate noise, the inverse of A can be written as

$$(DKD + \sigma^2 I)^{-1} = D^{-1}(K + \sigma^2 D^{-2})^{-1} D^{-1} \,, \tag{4.41}$$

which again can be computed with $\mathcal{O}(N)$ memory and $\mathcal{O}(N \log N)$ operations, since $K + \sigma^2 D^{-2}$ is Toeplitz and D is diagonal.

Finally, it would be useful to extend the proposed non-stationary models to multiple outputs, where a vector of basis functions $\boldsymbol{f}(x)$, representing Gaussian processes with spectral mixture (SM) kernels, is modulated by a whole matrix $W(x)$ of Gaussian processes with squared exponential kernels. Such a model could be used to fill in large missing segments of speech or music in channels of a stereo recording.

4.5 Alternate Kernels for Pattern Discovery

The spectral mixture (SM) kernel forms a powerful basis for all stationary kernels, but variations on the SM kernel could be useful alternatives. In section 4.4.2 we have explored variations on the SM kernel for modelling non-stationary data. One could construct many useful variations on the spectral mixture kernel.

Consider, for instance, a 1 component spectral mixture (SM) kernel for inputs $x \in \mathbb{R}^1$:

$$k_{\text{SM-1}}(\tau) = \exp\{-2\pi^2\tau^2\sigma^2\}\cos(2\pi\tau\mu)\,. \tag{4.42}$$

Following the derivation of the rational quadratic (RQ) kernel in section 2.4.4, we can integrate

$$\int_0^\infty k_{\text{SM-1}}(\tau|\sigma^2)p(\sigma^2)d\sigma^2 \tag{4.43}$$

to derive a kernel based on a scale mixture of 1-component SM kernels, with different length-scales (aka bandwidths σ^2). If we let

$$g(\gamma|\alpha,\beta) \propto \gamma^{\alpha-1}\exp(-\alpha\gamma/\beta)\,, \tag{4.44}$$

and we set $\gamma = \sigma^2$ and replace $p(\sigma^2)$ with $g(\gamma|\alpha,\beta)$, and perform the integral in Eq. (4.43), then we obtain

$$k_{\text{SM-RQ-1}} = (1 + \frac{\tau^2\sigma^2}{2\alpha})^{-\alpha}\cos(2\pi\tau\mu)\,, \tag{4.45}$$

after rescaling. One could use the kernel in Eq. (4.45) as an alternative basis for all stationary kernels. The result would be like a Gaussian mixture on the spectral density, except the mixture components would generally have heavier tails than Gaussians, with the tails controlled by the additional α hyperparameter. One could replace the exponential part of the SM kernel with a Matérn kernel for a similar effect.

4.6 Initialisation

In the experiments presented in this thesis, we typically initialise weight parameters as constants proportional to the standard deviation of the data, frequencies from a uniform distribution from 0 to Nyquist frequency (half the sampling rate of the data, or if the data are not regularly sampled, half the largest interval between input points), and length-scales from a truncated Gaussian distribution with mean proportional to the range of the inputs. In chapter 5, Figure 5.2, we show that extraneous weights in the SM kernel will shrink towards zero, as a proxy for model selection.

The SM kernel can be particularly sensitive to initialisation on small datasets, particularly those datasets which behave like pure mixtures of trigonometric functions, or contain obvious non-stationarities, like clear rising trends (including the CO_2 and airline data in sections 4.3.1 and 4.3.5). While the initialisation scheme outlined in the previous paragraph can still typically produce good results, initialisation can be significantly improved. For example, one can sample frequencies from the empirical spectral density, which is the Fourier transform of the empirical autocorrelation function (equivalently, the Fourier transform of the squared data). As an initialisation, one could also fit a mixture of Gaussians to the empirical spectral density, since the spectral mixture kernel has a dual representation as a mixture of Gaussians modelling a spectral density.

Note that the exponential terms in the SM kernel of Eq. (4.7) have an annealing effect on the marginal likelihood. The marginal likelihood is multimodal, largely because the longer the signal, the more various periodic components are able to fit parts of the data extremely well, and miss other parts entirely, resulting in multiple local optima. If the $v_q^{(p)}$ terms in the exponential part of the SM kernel are not initialized near zero, then these parameters effectively reduce the amount of data initially available, annealing the likelihood surface – ironing out local optima. Then, as the parameters are learned through marginal likelihood optimization, the final setting of hyperparameters is nearer a desired global optimum. For a description of a related simulated annealing method, see the end of section 3.9.1.

As we will see in chapter 5, the SM kernel is particularly robust to initialisation in the presence of large datasets. A demo of a GP with the SM kernel, with a variety of example initialisations, can be found at `http://mlg.eng.cam.ac.uk/andrew/pattern`.

4.7 Discussion

We have derived expressive closed form spectral mixture (SM) kernels and we have shown that these kernels, when used with Gaussian processes, can discover patterns in data and extrapolate over long ranges. The simplicity of these kernels is one of their strongest properties: they can be used as drop-in replacements for popular kernels such as the squared exponential kernel, with major benefits in expressiveness and performance, while retaining simple training and inference procedures.

Gaussian processes have proven themselves as powerful smoothing interpolators. We believe that pattern discovery and extrapolation is an exciting new direction for Gaussian processes. Bayesian nonparametric methods have the capacity to reflect the great complexity in the real world, and could be used to explain how biological intelligence is capable of making extraordinary generalisations from a small number of examples. When the large nonparametric support of a Gaussian process is combined with the ability to automatically discover patterns in data and extrapolate, we are a small step closer to developing truly intelligent agents, with applications in essentially any learning and prediction task.

In the future, one could compare human generalisation behaviour with modern statistical algorithms, on a wide range of learning tasks. For example, we can get an idea, in simple regression situations, of what a "human kernel" might look like, by asking a large number of participants to extrapolate various patterns. Understanding the discrepancies between human and machine generalization behaviour will help us continue to build more intelligent computational algorithms.

We would also like to further explore the non-stationary processes presented in this chapter, and combine these processes with scalable inference. These processes will have wide range applicability, with immediate applicability for reconstructing large missing segments of audio signals.

In the next chapter we further develop spectral mixure kernels for tractability with multidimensional patterns and a large number of training instances, in a method we refer to as *GPatt*. We exploit the kernels used with GPatt for scalable but *exact* inference. We particularly use GPatt to extrapolate large missing regions of images and videos.

Chapter 5

GPatt: Fast Multidimensional Pattern Extrapolation

Gaussian processes are typically used for smoothing and interpolation on small datasets. In this brief chapter we introduce a new Bayesian nonparametric framework – GPatt – enabling automatic pattern extrapolation with Gaussian processes on large multidimensional datasets. GPatt unifies and extends highly expressive spectral mixture (SM) kernels of the previous chapter with fast exact inference techniques. In particular, we develop related spectral mixture product (SMP) kernels, and exploit the structure in these kernels for fast and exact inference and hyperparameter learning. Specifically, inference and learning requires $\mathcal{O}(PN^{\frac{P+1}{P}})$ operations and $\mathcal{O}(PN^{\frac{2}{P}})$ storage, for N training points and P dimensional inputs, compared to the $\mathcal{O}(N^3)$ operations and $\mathcal{O}(N^2)$ storage for standard GP inference and learning.

Without human intervention – no hand crafting of kernel features, and no sophisticated initialisation procedures – we show that GPatt can solve practically important large scale pattern extrapolation, inpainting, video extrapolation, and kernel discovery problems, including a problem with 383,400 training points. We find that GPatt significantly outperforms popular alternative scalable Gaussian process methods in speed and accuracy. Moreover, we discover profound differences between each of these methods, suggesting expressive kernels, nonparametric representations, and the style are useful for modelling large scale multidimensional patterns. GPatt provides a fresh and highly general approach to multidimensional pattern discovery, enabling largely new applications, such as video extrapolation, which is difficult conventional methodology.

The material in this chapter and appendix D are closely based on the pre-print Wilson et al. (2013). Appendix D was largely contributed by Elad Gilboa.

5.1 Introduction

Kernel machines like Gaussian processes are typically unable to scale to large modern datasets. Methods to improve scalability usually involve simplifying assumptions, such as finite basis function expansions (Lázaro-Gredilla et al., 2010; Le et al., 2013; Rahimi and Recht, 2007; Williams and Seeger, 2001), or sparse approximations using pseudo (aka inducing) inputs (Hensman et al., 2013; Naish-Guzman and Holden, 2007; Quiñonero-Candela and Rasmussen, 2005; Seeger et al., 2003; Snelson and Ghahramani, 2006). The assumptions of these methods are often suitable for scaling popular kernel functions, but we will see that these assumptions are not as suitable for highly expressive kernels, particularly when a large number of training instances provide an opportunity to extract sophisticated structure from the data.

Indeed popular covariance kernels used with Gaussian processes are not often expressive enough to capture rich structure in data and perform extrapolation, prompting MacKay (1998) to ask whether we had "thrown out the baby with the bathwater". In general, the choice of kernel profoundly affects the performance of a Gaussian process – as much as the choice of architecture affects the performance of a neural network. Typically, Gaussian processes are used either as flexible statistical tools, where a human manually discovers structure in data and then hard codes that structure into a kernel, or with the popular Gaussian (squared exponential) or Matérn kernels. In either case, Gaussian processes are used as smoothing interpolators, only able to discover limited covariance structure. Likewise, multiple kernel learning (Gönen and Alpaydın, 2011) typically involves hand crafting combinations of Gaussian kernels for specialized applications, such as modelling low dimensional structure in high dimensional data, and is not intended for automatic pattern discovery and extrapolation.

In this chapter we propose a scalable and expressive Gaussian process framework, *GPatt*, for automatic pattern discovery and extrapolation on large multidimensional datasets. We assume the reader is familiar with the material in chapter 2, an introduction to Gaussian processes. In section 5.2 we introduce expressive interpretable kernels, which build off the spectral mixture kernels in chapter 4, but are especially structured for multidimensional inputs and for the fast exact inference and learning techniques we later introduce in section 5.3. These inference techniques work by exploiting the existing structure in the kernels of section 5.2 – and will also work with popular alternative kernels. These techniques relate to the recent inference methods of Saatchi (2011), but relax the full grid assumption made by these methods. This exact inference and learning costs $\mathcal{O}(PN^{\frac{P+1}{P}})$ computations and $\mathcal{O}(PN^{\frac{2}{P}})$ storage, for N datapoints and P input dimensions, compared to the standard $\mathcal{O}(N^3)$ computations and $\mathcal{O}(N^2)$ storage associated with a Cholesky decomposition.

In our experiments of section 5.4 we combine these fast inference techniques and expressive kernels to form GPatt. Our experiments emphasize that, although Gaussian processes are typically only used for smoothing and interpolation on small datasets, Gaussian process models can in fact be developed to automatically solve a variety of practically important large scale pattern extrapolation problems. GPatt is able to discover the underlying structure of an image, and extrapolate that structure across large distances, without human intervention – no hand crafting of kernel features, no sophisticated initialisation, and no exposure to similar images. We use GPatt to reconstruct large missing regions in pattern images, to restore a stained image, to reconstruct a natural scene by removing obstructions, and to discover a sophisticated 3D ground truth kernel from movie data. GPatt leverages a large number of training instances ($N > 10^5$) in many of these examples.

We find that GPatt significantly outperforms popular alternative Gaussian process methods on speed and accuracy stress tests. Furthermore, we discover profound behavioural differences between each of these methods, suggesting that expressive kernels, nonparametric representations[1], and exact inference are useful in combination for large scale multidimensional pattern extrapolation.

5.2 Spectral Mixture Product Kernel

The spectral mixture (SM) kernel (for 1D inputs) was derived in chapter 4 as

$$k_{\mathrm{SM}}(\tau) = \sum_{a=1}^{A} w_a^2 \exp\{-2\pi^2\tau^2\sigma_a^2\}\cos(2\pi\tau\mu_a)\,, \tag{5.1}$$

and applied solely to simple time series examples with a small number of datapoints.[2] In this chapter we extend this formulation for tractability with large datasets and multidimensional inputs.

The squared exponential kernel for multidimensional inputs, for example, decomposes as a product across input dimensions. This decomposition helps with computational tractability – limiting the number of hyperparameters in the model – and like stationarity, provides a bias that can help with learning. For higher dimensional inputs, $x \in \mathbb{R}^P$, we propose to leverage this useful product assumption for a spectral mixture product (SMP) kernel

$$k_{\mathrm{SMP}}(\tau|\boldsymbol{\theta}) = \prod_{p=1}^{P} k_{\mathrm{SM}}(\tau_p|\boldsymbol{\theta}_p)\,, \tag{5.2}$$

where τ_p is the p^{th} component of $\tau = x - x' \in \mathbb{R}^P$, $\boldsymbol{\theta}_p$ are the hyperparameters $\{\mu_a, \sigma_a^2, w_a^2\}_{a=1}^{A}$ of the p^{th} spectral mixture kernel in the product of Eq. (5.2), and $\boldsymbol{\theta} = \{\boldsymbol{\theta}_p\}_{p=1}^{P}$ are the hyperparameters of the SMP kernel. With enough components A, the SMP kernel of Eq. (5.2) can model any stationary product kernel to arbitrary precision, and is flexible even with a small number of components. We use SMP-A as shorthand for an SMP kernel with A components in each dimension (for a total of $3PA$ kernel hyperparameters and 1 noise hyperparameter).

[1]For a Gaussian process to be a Bayesian nonparametric model, its kernel must be derived from an infinite basis function expansion. The information capacity of such models grows with the data (Ghahramani, 2013).

[2]Here we parametrize weights as w^2 instead of w, so that w is roughly on the scale of the standard deviation of the data.

A GP with an SMP kernel is not a finite basis function method, but instead corresponds to a finite (PA component) mixture of infinite basis function expansions.

5.3 Fast Exact Inference

In this section we present algorithms which exploit the existing structure in the SMP kernels of section 5.2, and many other popular kernels, for significant savings in computation and memory, but with the same exact inference achieved with a Cholesky decomposition.

Gaussian process inference and learning requires evaluating $(K + \sigma^2 I)^{-1}\boldsymbol{y}$ and $\log|K + \sigma^2 I|$, for an $N \times N$ covariance matrix K, a vector of N datapoints $\boldsymbol{y}$, and noise variance σ^2, as in Equations (2.27) and (2.37), respectively. For this purpose, it is standard practice to take the Cholesky decomposition of $(K + \sigma^2 I)$ which requires $\mathcal{O}(N^3)$ computations and $\mathcal{O}(N^2)$ storage, for a dataset of size N. However, nearly any kernel imposes significant structure on K that is ignored by taking the Cholesky decomposition.

For example, we show how to exploit the structure of kernels that separate multiplicatively across P input dimensions,

$$k(x_i, x_j) = \prod_{p=1}^{P} k^p(x_i^p, x_j^p)\,, \tag{5.3}$$

to perform exact inference and hyperparameter learning in $\mathcal{O}(PN^{\frac{2}{P}})$ storage and $\mathcal{O}(PN^{\frac{P+1}{P}})$ operations, compared to the standard $\mathcal{O}(N^2)$ storage and $\mathcal{O}(N^3)$ operations. We first assume the inputs $x \in \mathcal{X}$ are on a multidimensional grid (section 5.3.1), meaning $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_P \subset \mathbb{R}^P$, and then relax this grid assumption[1] in section 5.3.2, so that the training data no longer need to be on a grid.

[1]Note the grid does not need to be regularly spaced.

5.3.1 Inputs on a Grid

Many real world applications are engineered for grid structure, including spatial statistics, sensor arrays, image analysis, and time sampling.

For a multiplicative (aka product) kernel and inputs on a grid, we find[1]

1. K is a Kronecker product of P matrices (a *Kronecker matrix*) which can undergo eigendecomposition into $QVQ^\top$ with only $\mathcal{O}(PN^{\frac{2}{P}})$ storage and $\mathcal{O}(PN^{\frac{3}{P}})$ computations (Saatchi, 2011), where Q is an orthogonal matrix of eigenvectors and V is a diagonal matrix of eigenvalues of K.[2]

2. The product of Kronecker matrices such as K, Q, or their inverses, with a vector $\boldsymbol{u}$, can be performed in $\mathcal{O}(PN^{\frac{P+1}{P}})$ operations.

Given the eigendecomposition of K as $QVQ^\top$, we can re-write $(K+\sigma^2 I)^{-1}\boldsymbol{y}$ and $\log|K+\sigma^2 I|$ in Eqs. (2.27) and (2.37) as

$$(K+\sigma^2 I)^{-1}\boldsymbol{y} = (QVQ^\top + \sigma^2 I)^{-1}\boldsymbol{y} \tag{5.4}$$

$$= Q(V+\sigma^2 I)^{-1}Q^\top \boldsymbol{y}\,, \tag{5.5}$$

and

$$\log|K+\sigma^2 I| = \log|QVQ^\top + \sigma^2 I| = \sum_{i=1}^{N} \log(\lambda_i + \sigma^2)\,, \tag{5.6}$$

where λ_i are the eigenvalues of K, which can be computed in $\mathcal{O}(PN^{\frac{3}{P}})$.

Thus we can evaluate the predictive distribution and marginal likelihood in Eqs. (2.27) and (2.37) to perform *exact* inference and hyperparameter learning, with $\mathcal{O}(PN^{\frac{2}{P}})$ storage and $\mathcal{O}(PN^{\frac{P+1}{P}})$ operations (assuming $P > 1$).

5.3.2 Missing Observations

Assuming we have a dataset of M observations which are not necessarily on a grid, we can form a complete grid using W imaginary observations, $\boldsymbol{y}_W \sim \mathcal{N}(\boldsymbol{f}_W, \epsilon^{-1} I_W)$,

[1]Details are in appendix D.

[2]The total number of datapoints $N = \prod_p |\mathcal{X}_p|$, where $|\mathcal{X}_p|$ is the cardinality of $\mathcal{X}_p$. For clarity of presentation, we assume each $|\mathcal{X}_p|$ has equal cardinality $N^{1/P}$.

$\epsilon \to 0$. The total observation vector $\boldsymbol{y} = [\boldsymbol{y}_M, \boldsymbol{y}_W]^\top$ has $N = M + W$ entries: $\boldsymbol{y} = \mathcal{N}(\boldsymbol{f}, D_N)$, where

$$D_N = \begin{bmatrix} D_M & 0 \\ 0 & \epsilon^{-1} I_W \end{bmatrix}, \tag{5.7}$$

and $D_M = \sigma^2 I_M$.[1] The imaginary observations $\boldsymbol{y}_W$ have *no corrupting effect* on inference: the moments of the resulting predictive distribution are exactly the same as for the standard predictive distribution in Eq. (2.27). E.g., $(K_N + D_N)^{-1}\boldsymbol{y} = (K_M + D_M)^{-1}\boldsymbol{y}_M$.[2]

In order to compute the predictive distribution of the Gaussian process, we use preconditioned conjugate gradients (PCG) (Atkinson, 2008) to compute $(K_N + D_N)^{-1}\,\boldsymbol{y}$. In particular, we use the preconditioning matrix $C = D_N^{-1/2}$ to solve

$$C^\top (K_N + D_N)\, C\boldsymbol{z} = C^\top \boldsymbol{y}\,. \tag{5.8}$$

The preconditioning matrix C speeds up convergence by ignoring the imaginary observations $\boldsymbol{y}_W$. Exploiting the fast multiplication of Kronecker matrices, PCG takes $\mathcal{O}(JPN^{\frac{P+1}{P}})$ total operations (where the number of iterations $J \ll N$) to compute $(K_N + D_N)^{-1}\,\boldsymbol{y}$, which allows for exact inference. J is typically small for convergence within machine precision, and is essentially independent of N, but is more sensitive to the conditioning of $C^\top (K_N + D_N)$.

For learning (hyperparameter training) we must evaluate the marginal likelihood of Eq. (2.37). We cannot efficiently decompose $K_M + D_M$ to compute the $\log |K_M + D_M|$ complexity penalty in the marginal likelihood, because K_M is not a Kronecker matrix, since we have an incomplete grid. We approximate the complexity penalty as

$$\log |K_M + D_M| = \sum_{i=1}^{M} \log(\lambda_i^M + \sigma^2) \tag{5.9}$$

$$\approx \sum_{i=1}^{M} \log(\tilde{\lambda}_i^M + \sigma^2)\,, \tag{5.10}$$

[1]We sometimes use subscripts on matrices to emphasize their dimensionality: e.g., D_N, D_M, and I_W are respectively $N \times N$, $M \times M$, and $W \times W$ matrices.

[2]See appendix D4 for a proof.

where we approximate the eigenvalues λ_i^M of K_M using the eigenvalues of K_N such that $\tilde{\lambda}_i^M = \frac{M}{N}\lambda_i^N$ for $i = 1, \ldots, M$, which is a particularly good approximation for large M (e.g. $M > 1000$) (Williams and Seeger, 2001). We emphasize that only the log determinant (complexity penalty) term in the marginal likelihood undergoes a small approximation, and inference remains exact.

All remaining terms in the marginal likelihood of Eq. (2.37) can be computed exactly and efficiently using PCG. The total runtime cost of hyperparameter learning and exact inference with an incomplete grid is thus $\mathcal{O}(PN^{\frac{P+1}{P}})$. In image problems, for example, $P = 2$, and so the runtime complexity reduces to $\mathcal{O}(N^{1.5})$.

We emphasize that although the proposed inference can handle non-grid data, this inference is most suited to inputs where there is some grid structure – images, video, spatial statistics, etc. If there is no such grid structure (e.g., none of the training data fall onto a grid), then the computational expense necessary to augment the data with imaginary grid observations is exponential in the number of input dimensions and can be prohibitive for e.g., $P > 5$. If there is such grid structure, however, the computational expense in handling N datapoints decreases with P.

5.4 Experiments

In our experiments we combine the SMP kernel of Eq. (5.2) with the fast exact inference and learning procedures of section 5.3, in a GP method we henceforth call GPatt[1], to perform extrapolation on a variety of sophisticated patterns embedded in large datasets.

We contrast GPatt with many alternative Gaussian process kernel methods. In particular, we compare to the recent sparse spectrum Gaussian process regression (SSGP) (Lázaro-Gredilla et al., 2010) method, which provides fast and flexible kernel learning. SSGP models the kernel spectrum (spectral density) as a sum of point masses, such that SSGP is a finite basis function model, with as many basis functions as there are spectral point masses. SSGP is similar to the recent models

[1]We write *GPatt-A* when GPatt uses an SMP-A kernel.

of Le et al. (2013) and Rahimi and Recht (2007), except it learns the locations of the point masses through marginal likelihood optimization. We use the SSGP implementation provided by the authors at `http://www.tsc.uc3m.es/~miguel/downloads.php`.

To further test the importance of the fast inference (section 5.3) used in GPatt, we compare to a GP which uses the SMP kernel of section 5.2 but with the popular fast FITC (Naish-Guzman and Holden, 2007; Snelson and Ghahramani, 2006) inference, implemented in GPML.[1] We also compare to Gaussian processes with the popular squared exponential (SE), rational quadratic (RQ) and Matérn (MA) (with 3 degrees of freedom) kernels, catalogued in Rasmussen and Williams (2006), respectively for smooth, multi-scale, and finitely differentiable functions. Since Gaussian processes with these kernels cannot scale to the large datasets we consider, we combine these kernels with the same fast inference techniques that we use with GPatt, to enable a comparison.[2]

Moreover, we stress test each of these methods, in terms of speed and accuracy, as a function of available data and extrapolation range, number of components, and noise. Experiments were run on a 64bit PC, with 8GB RAM and a 2.8 GHz Intel i7 processor.

In all experiments we assume Gaussian noise, so that we can express the likelihood of the data $p(\boldsymbol{y}|\boldsymbol{\theta})$ solely as a function of kernel hyperparameters $\boldsymbol{\theta}$. To learn $\boldsymbol{\theta}$ we optimize the marginal likelihood using BFGS. We use a simple initialisation scheme: any frequencies $\{\mu_a\}$ are drawn from a uniform distribution from 0 to the Nyquist frequency (1/2 the sampling rate), length-scales $\{1/\sigma_a\}$ from a truncated Gaussian distribution, with mean proportional to the range of the data, and weights $\{w_a\}$ are initialised as the empirical standard deviation of the data divided by the number of components used in the model. In general, we find GPatt is robust to initialisation.

[1]`http://www.gaussianprocess.org/gpml`

[2]We also considered comparing to Duvenaud et al. (2013), but this model is intractable for the datasets we considered and is not structured for the fast inference of section 5.3, having been designed for a different purpose.

This range of tests allows us to separately understand the effects of the SMP kernel and proposed inference methods of section 5.3; we will show that both are required for good performance.

5.4.1 Extrapolating a Metal Tread Plate Pattern

We extrapolate the missing region, shown in Figure 5.1a, on a real metal tread plate texture. There are 12675 training instances (Figure 5.1a), and 4225 test instances (Figure 5.1b). The inputs are pixel locations $x \in \mathbb{R}^2$ ($P = 2$), and the outputs are pixel intensities. The full pattern is shown in Figure 5.1c. This texture contains shadows and subtle irregularities, no two identical diagonal markings, and patterns that have correlations across both input dimensions.

To reconstruct the missing region, as well as the training region, we use GPatt with 30 components for the SMP kernel of Eq. (5.2) in each dimension (GPatt-30). The GPatt reconstruction shown in Figure 5.1d is as plausible as the true full pattern shown in Figure 5.1c, and largely automatic. Without human intervention – no hand crafting of kernel features to suit this image, no sophisticated initialisation, and no exposure to similar images – GPatt has discovered the underlying structure of this image and extrapolated that structure across a large missing region, even though the structure of this pattern is not independent across the two spatial input dimensions. Indeed the separability of the SMP kernel represents only a soft prior assumption, and does not rule out posterior correlations between input dimensions. In general, we have found the inductive bias of separability useful – both for the kronecker structure it allows us to exploit, and for the learning rate of the method – and strong posterior correlations between input dimensions can still easily be discovered with the typical amount of data present in an image ($N > 10^4$).

The reconstruction in Figure 5.1e was produced with SSGP, using 500 basis functions. In principle SSGP can model any spectral density (and thus any stationary kernel) with infinitely many components (basis functions). However, for computational reasons the model can only accommodate finitely many components in practice, and since these components are point masses (in frequency space), each component has highly limited expressive power. Moreover, with many components SSGP experiences practical difficulties regarding initialisation, over-fitting, and computation time (scaling quadratically with the number of basis functions). Although SSGP does discover some interesting structure (a diagonal pattern), and has equal training and test performance, it is unable to capture enough information for a convincing reconstruction, and we did not find that more basis functions improved performance. Likewise, FITC with an SMP-30 kernel and 500 pseudo-inputs cannot capture the necessary information to interpolate or extrapolate. We note FITC and SSGP-500 respectively took 2 days and 1 hour to run on this example, compared to GPatt which took under 5 minutes.

GPs with SE, MA, and RQ kernels are all truly Bayesian nonparametric models – these kernels are derived from infinite basis function expansions. Therefore, as seen in Figure 5.1 g), h), i), these methods are completely able to capture the information in the training region; however, these kernels do not have the proper structure to reasonably extrapolate across the missing region – they simply act as smoothing filters. We note that this comparison is only possible because these GPs are using the fast exact inference techniques in section 5.3.

Overall, these results indicate that both expressive nonparametric kernels, such as the SMP kernel, and the specific fast inference in section 5.3, are needed to be able to extrapolate patterns in these images.

We note that the SMP-30 kernel used with GPatt has more components than needed for this problem. However, as shown in Figure 5.2, if the model is over-specified, the complexity penalty in the marginal likelihood shrinks the weights ($\{w_a\}$ in Eq. (5.1)) of extraneous components, as a proxy for model selection – an effect similar to *automatic relevance determination* (MacKay, 1994). As per Eq. (5.6), this complexity penalty can be written as a sum of log eigenvalues of a covariance matrix K. Components which do not significantly contribute to model fit will therefore be automatically pruned, as shrinking the weights decreases the eigenvalues of K and thus minimizes the complexity penalty. This weight shrinking helps both mitigate the effects of model overspecification and helps indicate

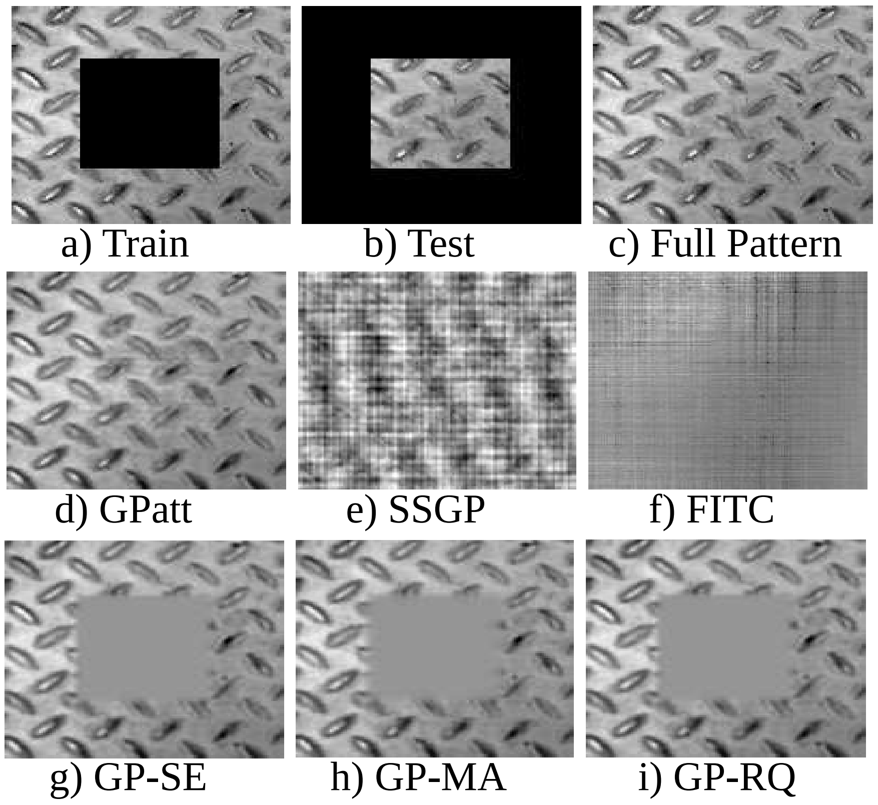


Figure 5.1: Extrapolation on a metal tread plate pattern. Missing data are shown in black. a) Training region (12675 points), b) Testing region (4225 points), c) Full tread plate pattern, d) GPatt-30, e) SSGP with 500 basis functions, f) FITC with 500 pseudo inputs, and the SMP-30 kernel, and GPs with the fast exact inference in section 5.3.1, and g) squared exponential (SE), h) Matérn (MA), and i) rational quadratic (RQ) kernels.

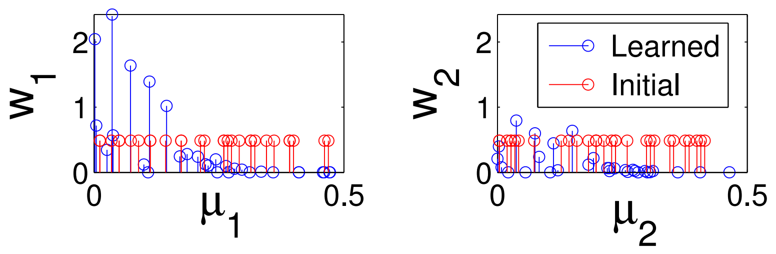


Figure 5.2: Automatic model selection in GPatt. Initial and learned weight and frequency parameters of GPatt-30, for each input dimension (a dimension is represented in each panel), on the metal tread plate pattern of Figure 5.1. GPatt-30 is overspecified for this pattern. During training, weights of extraneous components automatically shrink to zero, which helps indicate whether the model is overspecified, and helps mitigate the effects of model overspecification. Of the 30 initial components in each dimension, 15 are near zero after training.

whether the model is overspecified. In the following stress tests we find that GPatt scales efficiently with the number of components in its SMP kernel.

5.4.2 Stress Tests

We stress test GPatt and alternative methods in terms of speed and accuracy, with varying datasizes, extrapolation ranges, basis functions, pseudo inputs, and components. We assess accuracy using standardised mean square error (SMSE) and mean standardized log loss (MSLL) (a scaled negative log likelihood), as defined in Rasmussen and Williams (2006) on page 23. Using the empirical mean and variance to fit the data would give an SMSE and MSLL of 1 and 0 respectively. Smaller SMSE and more negative MSLL values correspond to better fits of the data.

The runtime stress test in Figure 5.3a shows that the number of components used in GPatt does not significantly affect runtime, and that GPatt is much faster than FITC (using 500 pseudo inputs) and SSGP (using 90 or 500 basis functions), even with 100 components (601 kernel hyperparameters). The slope of each curve roughly indicates the asymptotic scaling of each method. In this experiment, the standard GP (with SE kernel) has a slope of 2.9, which is close to the cubic scaling we expect. All other curves have a slope of 1 ± 0.1, indicating linear

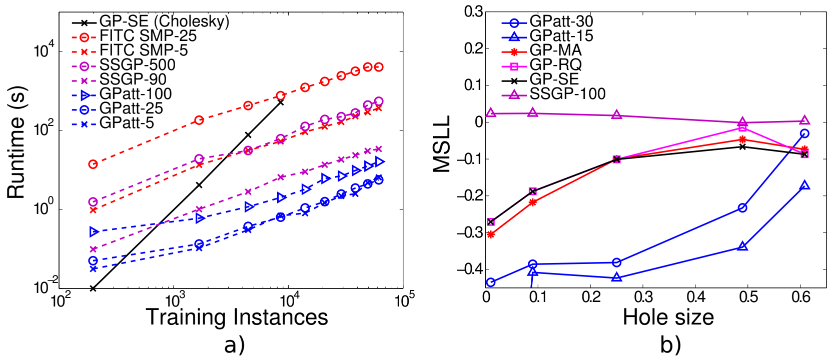
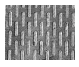
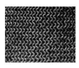


Figure 5.3: Stress tests. a) **Runtime Stress Test**. We show the runtimes in seconds, as a function of training instances, for evaluating the log marginal likelihood, and any relevant derivatives, for a standard GP with SE kernel (as implemented in GPML), FITC with 500 pseudo-inputs and SMP-25 and SMP-5 kernels, SSGP with 90 and 500 basis functions, and GPatt-100, GPatt-25, and GPatt-5. Runtimes are for a 64bit PC, with 8GB RAM and a 2.8 GHz Intel i7 processor, on the cone pattern ($P = 2$), shown in Figure 5.4. The ratio of training inputs to the sum of imaginary and training inputs for GPatt (section 5.3.2) is 0.4 and 0.6 for the smallest two training sizes, and 0.7 for all other training sets. b) **Accuracy Stress Test**. MSLL as a function of holesize on the metal pattern of Figure 5.1. The values on the horizontal axis represent the fraction of missing (testing) data from the full pattern (for comparison Fig 5.1a has 25% missing data). We compare GPatt-30 and GPatt-15 with GPs with SE, MA, and RQ kernels (and the inference of section 5.3), and SSGP with 100 basis functions. The MSLL for GPatt-15 at a holesize of 0.01 is -1.5886.

scaling with the number of training instances. However, FITC and SSGP are used here with a *fixed* number of pseudo inputs and basis functions. More pseudo inputs and basis functions should be used when there are more training instances – and these methods scale quadratically with pseudo inputs and basis functions for a fixed number of training instances. GPatt, on the other hand, can scale linearly in runtime as a function of training size, without any deterioration in performance. Furthermore, the big gaps between each curve – the fixed 1-2 orders of magnitude GPatt outperforms alternatives – are as practically important as asymptotic scaling.

The accuracy stress test in Figure 5.3b shows extrapolation (MSLL) performance on the metal tread plate pattern of Figure 5.1c with varying holesizes, running from 0% to 60% missing data for testing (for comparison the hole shown in Figure

Table 5.1: We compare the test performance of GPatt-30 with SSGP (using 100 basis functions), and GPs using squared exponential (SE), Matérn (MA), and rational quadratic (RQ) kernels, combined with the inference of section 5.4.2, on patterns with a train test split as in the metal treadplate pattern of Figure 5.1.

	GPatt	SSGP	SE	MA	RQ
	Rubber mat	(train = 12675, test = 4225)			
SMSE	**0.31**	0.65	0.97	0.86	0.89
MSLL	**−0.57**	−0.21	0.14	−0.069	0.039
	Tread plate	(train = 12675, test = 4225)			
SMSE	**0.45**	1.06	0.895	0.881	0.896
MSLL	**−0.38**	0.018	−0.101	−0.1	−0.101
	Pores	(train = 12675, test = 4225)			
SMSE	**0.0038**	1.04	0.89	0.88	0.88
MSLL	**−2.8**	−0.024	−0.021	−0.024	−0.048
	Wood	(train = 14259, test = 4941)			
SMSE	**0.015**	0.19	0.64	0.43	0.077
MSLL	**−1.4**	−0.80	1.6	1.6	0.77
	Chain mail	(train = 14101, test = 4779)			
SMSE	**0.79**	1.1	1.1	0.99	0.97
MSLL	**−0.052**	0.036	1.6	0.26	−0.0025

5.1a is for 25% missing data). GPs with SE, RQ, and MA kernels (and the fast inference of section 5.3) all steadily increase in error as a function of holesize. Conversely, SSGP does not increase in error as a function of holesize – with finite basis functions SSGP cannot extract as much information from larger datasets as the alternatives. GPatt performs well relative to the other methods, even with a small number of components. GPatt is particularly able to exploit the extra information in additional training instances: only when the holesize is so large that over 60% of the data are missing does GPatt's performance degrade to the same level as alternative methods.

In Table 5.1 we compare the test performance of GPatt with SSGP, and GPs using SE, MA, and RQ kernels, for extrapolating five different patterns, with the same train test split as for the tread plate pattern in Figure 5.1. All patterns for the stress tests in this section are shown in large in Figure 5.4.

GPatt consistently has the lowest standardized mean squared error (SMSE), and mean standardized log loss (MSLL). Note that many of these datasets are sophisticated patterns, containing intricate details and subtleties which are not strictly periodic, such as lighting irregularities, metal impurities, etc. Indeed SSGP has a periodic kernel (unlike the SMP kernel which is not strictly periodic), and is capable of modelling multiple periodic components, but does not perform as well as GPatt on these examples.

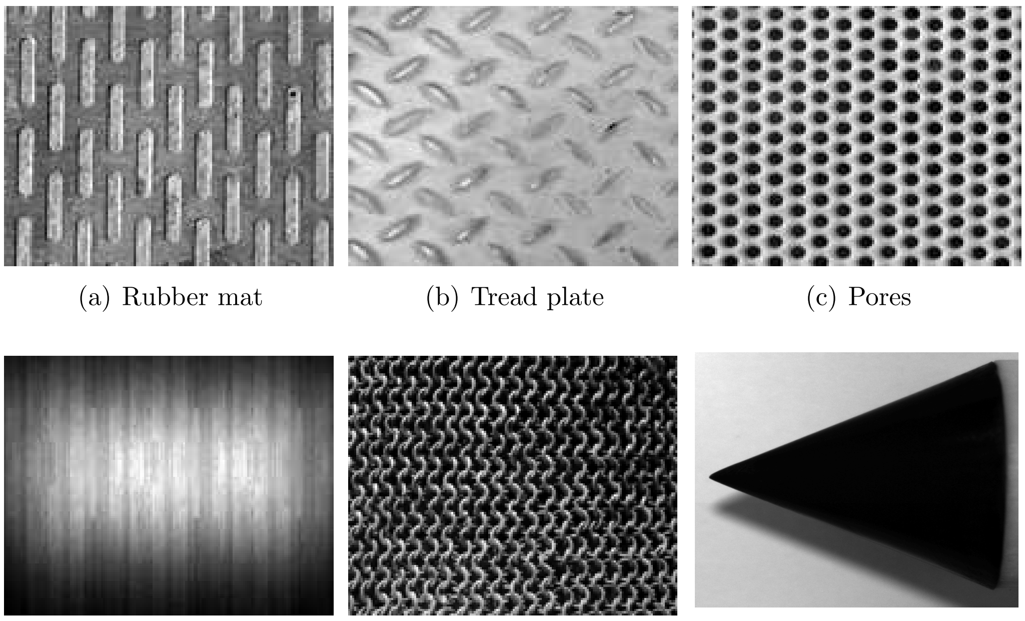
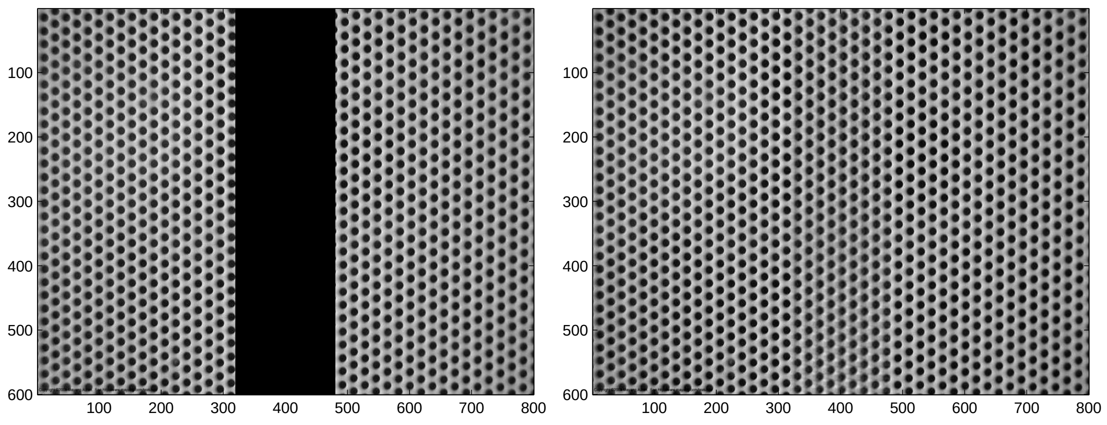


Figure 5.4: Images used for stress tests. Figures a) through e) show the textures used in the accuracy comparison of Table 5.1. Figure e) is the cone image which was used for the runtime analysis shown in Figure 5.3a

We end this section with a particularly large example, where we use GPatt-10 to perform learning and exact inference on the *Pores* pattern, with 383400 training points, to extrapolate a large missing region with 96600 test points. The SMSE is



(a) Training Data (b) Predictions

Figure 5.5: GPatt extrapolation on a particularly large multidimensional dataset. a) Training region (383400 points), b) GPatt-10 reconstruction of the missing region.

0.077, and the total runtime was 2800 seconds. Images of the successful extrapolation are shown in Figure 5.5.

5.4.3 Wallpaper and Scene Reconstruction

Although GPatt is a general purpose regression method, it can also be used for inpainting: image restoration, object removal, etc.

We first consider a wallpaper image stained by a black apple mark, shown in the first row of Figure 5.6. To remove the stain, we apply a mask and then separate the image into its three channels (red, green, and blue). This results in 15047 pixels in each channel for training. In each channel we ran GPatt using SMP-30. We then combined the results from each channel to restore the image without any stain, which is particularly impressive given the subtleties in the pattern and lighting.

In our next example, we wish to reconstruct a natural scene obscured by a prominent rooftop, shown in the second row of Figure 5.6. By applying a mask, and following the same procedure as for the stain, this time with 32269 pixels in each channel for training, GPatt reconstructs the scene without the rooftop. This reconstruction captures subtle details, such as waves in the ocean, even though only one image was used for training.

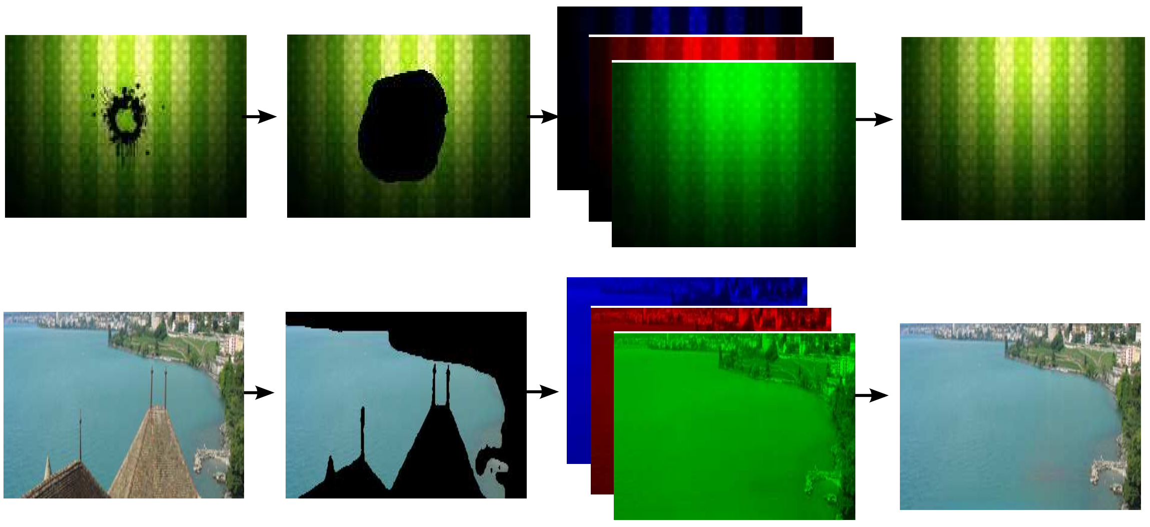


Figure 5.6: Image inpainting with GPatt. From left to right: A mask is applied to the original image, GPatt extrapolates the mask region in each of the three (red, blue, green) image channels, and the results are joined to produce the restored image. Top row: Removing a stain (train: 15047×3). Bottom row: Removing a rooftop to restore a natural scene (train: 32269×3). The coast is masked during training and we do not attempt to extrapolate it in testing.

Inpainting is an active area of research in computer vision. There are several distinct approaches to inpainting. The popular patch based methods (Criminisi et al., 2004; Efros and Leung, 1999) recursively fill in pixels (or patches of pixels) from the outside of a gap in the image. E.g., a known pixel (pixel α) is selected to fill in a missing pixel on the boundary of the gap (pixel β), such that the neighbourhoods of α and β are most similar. Markov texture models (Cross and Jain, 1983; Derin and Elliott, 1987; Geman and Geman, 1984; Hassner and Sklansky, 1980) are also popular. These models are based on Markov random fields (MRF), which characterize a model based on statistical interactions within local neighbourhoods. Portilla and Simoncelli (2000), for example, propose a parametric model with Markov statistical descriptors which are based on pairs of wavelet coefficients at adjacent spatial locations, orientations, and scales. Although GPatt is nonparametric, and is not a MRF, the discussion in Portilla and Simoncelli (2000) leads us to believe that GPatt could be further improved, for the purpose of texture modelling and inpainting, by extending the model to account for higher order statistics.

Many inpainting methods are highly application specific, involve significant hand crafting of features and human intervention, and are focused on making the reconstruction look good to a human. For example, Portilla and Simoncelli (2000) write that the work of Julesz (1962) established validation of texture models through "human perceptual comparison". Similarly Hays and Efros (2008) maintain that development and assessment of image models "relies on the studies of human visual perception".

Hays and Efros (2008) cite the rooftop problem (in the second panel of the lower row of Figure 5.6) as a particularly challenging inpainting problem, which they conjecture cannot be satisfactorly solved from the training data in that one image alone. They propose an algorithm (similar to a patch based method) which uses information from multiple similar images for inpainting on a particular image. After exposing their algorithm to a repository of similar images – in this case, various harbours – Hays and Efros (2008) provide a reconstruction of the image in Figure 5.6 (bottom left) without the rooftop. The results highlight the stark conceptual differences between GPatt and their algorithm. For example, Hays and Efros (2008) placed boats into the water, after having removed the rooftop. But there is nothing compelling in the original image to support boats being in the water; and even if we were to think that boats would be there, we couldn't possibly know the structure, orientations, or locations of these boats. Such an algorithm would therefore have poor performance in terms of squared loss or predictive likelihood. Indeed, general purpose regression algorithms, such as GPatt, simply aim to make accurate predictions at test input locations from training data alone, and this objective can even be at cross purposes with making the reconstruction look good to a human.

5.4.4 Recovering Complex 3D Kernels and Video Extrapolation

With a relatively small number of components, GPatt is able to accurately recover a wide range of product kernels. To test GPatt's ability to recover ground truth kernels, we simulate a $50 \times 50 \times 50$ movie of data (e.g. two spatial input dimensions, one temporal) using a GP with kernel $k = k_1 k_2 k_3$ (each component kernel in this product operates on a different input dimension), where $k_1 = k_{\text{SE}} + k_{\text{SE}} \times k_{\text{PER}}$, $k_2 = k_{\text{MA}} \times k_{\text{PER}} + k_{\text{MA}} \times k_{\text{PER}}$, and $k_3 = (k_{\text{RQ}} + k_{\text{PER}}) \times k_{\text{PER}} + k_{\text{SE}}$. ($k_{\text{PER}}(\tau) = \exp[-2\sin^2(\pi\,\tau\,\omega)/\ell^2]$, $\tau = x - x'$). We use 5 consecutive 50×50 slices for testing, leaving a large number $N = 112500$ of training points. In this case, the big datasize is helpful: the more training instances, the more information to learn the true generating kernels. Moreover, GPatt-20 is able to reconstruct these complex out of class kernels in under 10 minutes. We compare the learned SMP-20 kernel with the true generating kernels in Figure 5.8. In Figure 5.7, we show true and predicted frames from the movie.

The predictions in Figure 5.7 are an example of *video extrapolation* – predicting a sequence of frames that are entirely missing from the movie. GPatt can be directly applied to such problems, as it is a model for general purpose pattern extrapolation, rather than, for instance, a specialized inpainting algorithm. From the perspective of GPatt, video extrapolation and image inpainting are fundamentally the same task, except video extrapolation has one additional input dimension, which is easily included in GPatt's kernel. The kernel can learn both the temporal and spatial correlation structure. Moving to yet higher dimensional pattern extrapolation problems would proceed similarly, even if each input dimension had different correlation structures. For example, GPatt could be applied to extrapolating three dimensional representations through time, a four dimensional problem.

On the other hand, conventional image inpainting algorithms, particularly patch based methods, cannot be applied to video extrapolation, because they do not account for temporal correlations. There has been some work on special instances of video extrapolation, for example, i) when there is a moving object on a stationary background (Patwardhan et al., 2005), ii) relatively small missing patches from frame(s) in a video rather than entirely missing frames (Granados et al., 2012) (called *video inpainting*), and iii) simulating repetitive 2D patterns (in larger regions than in video inpainting), such as a flickering candle or a waving flag (*video textures*) (Schödl et al., 2000). Like inpainting methods, these methods tend to be highly specialized to a given application. Overall, video extrapolation is thought to be an exceptionally difficult and unresolved problem (Guillemot and Le Meur, 2014; Moran, 2009). Indeed video restoration is presently performed by "professionals in a manual fashion, which is not only painstaking and slow but also rather expensive. Therefore any means of automation would certainly benefit both commerical organizations and private individuals" (Moran, 2009). Moreover, Guillemot and Le Meur (2014) writes that inpainting algorithms have "limited direct applicability for video inpainting, which remains an open problem, despite preliminary solutions making assumptions in terms of moving objects or camera motion".

In the future, it would be interesting to explore GPatt's ability to solve video extrapolation problems, and even higher dimensional pattern extrapolation problems. I believe the generality of this approach to pattern extrapolation could open up entirely new application areas.

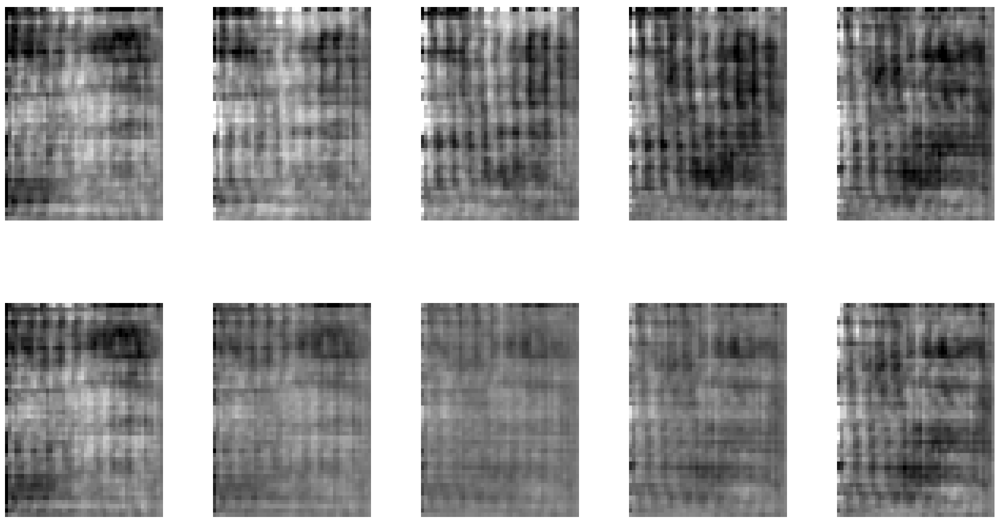


Figure 5.7: Using GPatt to recover 5 consecutive slices from a movie. All slices are missing from training data (e.g., these are not 1 step ahead forecasts). Top row: true slices take from the middle of the movie. Bottom row: inferred slices using GPatt-20.

5.5 Discussion

Gaussian processes are often used for smoothing and interpolation on small datasets. However, we believe that Bayesian nonparametric models are naturally suited to

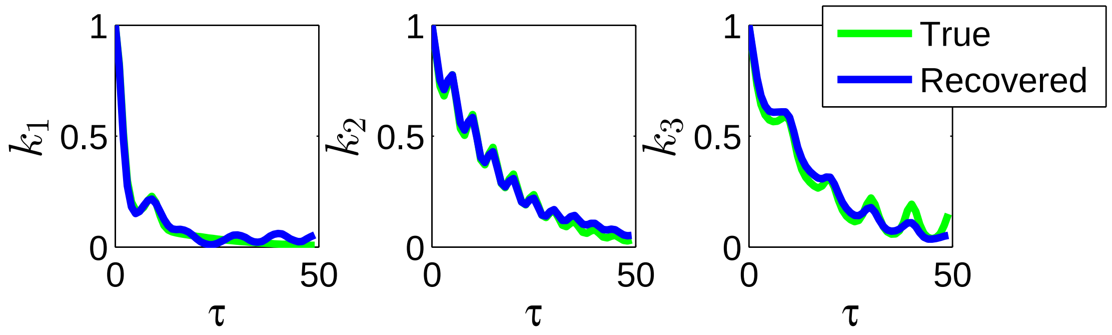


Figure 5.8: Recovering sophisticated product kernels from video data using GPatt. A product of three kernels (shown in green) was used to generate a movie of 112500 points with $P = 3$ input variables. From this data, GPatt-20 reconstructs these component kernels (the learned SMP-20 kernel is shown in blue). All kernels are a function of $\tau = x - x'$. For clarity of presentation, each kernel has been scaled by $k(0)$.

pattern extrapolation on large multidimensional datasets, where extra training instances can provide extra opportunities to learn additional structure in data.

The support and inductive biases of a Gaussian process are naturally encoded in a covariance kernel. A covariance kernel must always have some structure to reflect these inductive biases; and that structure can, in principle, be exploited for scalable and exact inference, without the need for simplifying approximations. Such models could play a role in a new era of machine learning, where models are expressive and scalable, but also interpretable and manageable, with simple exact learning and inference procedures.

We hope to make a small step in this direction with GPatt, a Gaussian process based Bayesian nonparametric framework for automatic pattern discovery on large multidimensional datasets, with scalable and exact inference procedures. Without human intervention – no sophisticated initialisation, or hand crafting of kernel features – GPatt has been used to accurately and quickly extrapolate large missing regions on a variety of patterns.

Chapter 6

Discussion

Pattern discovery and extrapolation is at the heart of probabilistic modelling. This thesis shows that one can develop powerful Bayesian nonparametric Gaussian process models which enable pattern discovery and extrapolation, and greatly improve predictions, with expressive covariance kernels. This expressive power does not necessarily come at an additional computational expense, as the structure in the proposed models can be exploited for fast and exact inference and learning procedures. The high level ideas presented in this thesis make it easy to develop new models with similar properties, which can learn structure in data that was previously undiscoverable. I summarize some of these high-level ideas and applications:

1. The ability for a model to discover structure in data, and extrapolate that structure to new situations, is determined by its support and inductive biases.
2. One should always develop models with the largest support possible, and distribute that support carefully for inductive biases that allow as much information as possible to be extracted from the data.
3. Bayesian nonparametric models are able to accommodate priors with large support and detailed inductive biases, and are thus naturally suited to pattern discovery and extrapolation.
4. Although Bayesian nonparametric models are typically applied to small datasets,

they are ideally suited to large datasets, because when there is more data, there is typically more information available to learn detailed structure.

5. Gaussian processes are rich distributions over functions, which provide a Bayesian nonparametric approach to smoothing and interpolation. The covariance kernel determines the support and inductive biases of a Gaussian process. Expressive covariance kernels can be developed to enable pattern discovery and extrapolation.

6. Spectral density modelling is an easy and intuitive recipe for constructing new covariance kernels. Likewise, one can develop expressive kernels by combining the nonparametric flexibility of Gaussian processes with certain structural properties, e.g., adaptive basis functions, of neural networks. The structure behind an adaptive network, for example, can be viewed as an undirected graphical model between latent adaptive basis functions and outputs, where the connections in this model are themselves functions of the inputs.

7. A model, or more specifically a covariance kernel, must always have some structure to reflect inductive biases which allow a model to extract useful information from data. In principle, this existing structure can be exploited for scalable inference and learning techniques, without additional simplifying assumptions.

The models in this thesis were applied to problems in econometrics, NMR spectroscopy, geo-statistics, and large-scale image inpainting, texture extrapolation, video extrapolation, and kernel discovery. In these applications these models have proven to be scalable and with greatly enhanced predictive performance over the alternatives: the extra structure we are learning is an important part of real data. I believe we will soon enter a new era of machine learning, where models are highly expressive, but also interpretable and manageable, and are scalable through simple inference and learning procedures which exploit existing model structure. Such models will help automate pattern discovery, learning, and decision making, with applications in essentially any prediction task. I hope to contribute to this direction with the models in this thesis.

Appendix A

Time Series

A1 Introduction

Time is a special input variable because it indicates the direction of causality. In this dissertation, we introduce models that can easily take any arbitrary variable as input, but we often compare with *time series* models in econometrics. There are four types of time series: discrete time discrete space, discrete time continuous space, continuous time discrete space, and continuous time continuous space. Many popular time series models (AR, MA, ARMA, ARCH, . . .) are in the discrete time continuous space class. In this section we introduce some of these models, after we have defined relevant terminology. This section is mostly a condensed reference; for more detail, see Shumway and Stoffer (2006) or Tsay (2002).

A time series is a stochastic process that describes the evolution of a random variable over time. Equivalently,

Definition A1.1. *A time series series model defines a joint probability distribution over any collection of random variables $x(t_1), \dots, x(t_k)$ indexed by time.*

A time series $x(t)$[1] is partly characterized by its *mean function* $\mu(t) = \mathbb{E}[x(t)]$, and *autocovariance function* $k(s,t) = \mathrm{cov}(x(s), x(t)) = \mathbb{E}[(x(s) - \mu(s))(x(t) - \mu(t)]$. The

[1]In our descriptions, we use the notation $x(t)$ and x_t interchangeably, to mean x evaluated at (or indexed by) the input variable t.

autocorrelation function (referred to as the ACF) is the normalised autocovariance function, $\rho(s,t) = \frac{k(s,t)}{\sqrt{k(s,s)k(t,t)}}$.

A *strictly stationary* time series is one for which the probability distribution of every collection of values $\{x(t_1), x(t_2), \dots, x(t_k)\}$ is the same as $\{x(t_1+a), x(t_2+a), \dots, x(t_k+a)\}$ for all feasible a. A *weakly stationary* time series $x(t)$ has $\mu(t) = c$ for all t, and $k(s,t) = k(s+a, t+a)$ for all feasible a. In other words, the *lag-l autocovariance*, $\gamma_l = k(t, t-l)$, is only a function of l, not t. For a weakly stationary series, the *lag-l ACF* is $\rho_l = \frac{\mathrm{cov}(x(t), x(t-l))}{\sqrt{\mathrm{Var}(x(t))\mathrm{Var}(x(t-l))}} = \frac{\gamma_l}{\mathrm{Var}(x(t))} = \frac{\gamma_l}{\gamma_0}$.

A *white noise* time series $\epsilon(t)$ has a zero mean function and constant variance. For $l \neq 0$, the ACF of this series is zero: each random variable at time t is iid. Each $\epsilon(t_i)$ is called an *error term* or *random shock.*

In an autoregressive AR(1) model, the value of the variable $x(t)$ deterministically depends on the value of $x(t-1)$ plus white noise. For example, if the white noise $\epsilon(t)$ has a Bernoulli distribution, such that $\epsilon(t) = 1$ with probability 0.5, and $\epsilon(t) = -1$ with probability 0.5, then this AR(1) process is a symmetric random walk. The general AR(p) process is

$$x(t) = a_0 + \sum_{i=1}^{p} a_i x(t-i) + \epsilon(t)\,, \tag{A.1}$$

where $\epsilon(t)$ is a white noise series. If $\epsilon(t)$ is Gaussian, then Eq. (A.1) is an example of a discrete time autoregressive Gaussian process. In chapter 4 we give an example of such a a Gaussian process which has a covariance function with negative covariances.

To choose the order of an AR model, one can use the *partial autocorrelation function* (PACF). Fitting the following AR models to data, using least squares,

$$x_t = a_{0,1} + a_{1,1}x_{t-1} + \epsilon_{1t} \tag{A.2}$$

$$x_t = a_{0,2} + a_{1,2}x_{t-1} + a_{2,2}x_{t-2} + \epsilon_{2t} \tag{A.3}$$

$$x_t = a_{0,3} + a_{1,3}x_{t-1} + a_{2,3}x_{t-2} + a_{3,3}x_{t-3} + \epsilon_{3t} \tag{A.4}$$

$$\vdots$$

the least squares estimate $\hat{a}_{j,j}$ is the *lag-j* PACF of $x(t)$. It can be shown that $\hat{a}_{j,j}$ converges to zero for all $j > p$, for an AR(p) process.

The moving average MA(q) process is

$$x(t) = b_0 + \sum_{i=1}^{q} b_i \epsilon(t-i)\,, \tag{A.5}$$

where $\epsilon(t)$ is a white noise series. While the *lag-l* ACF of an AR series is never zero, the *lag-l* ACF of an MA series is zero for $l > q$. For a given data set, the empirical ACF may suggest that the MA model is superior to the AR model. And sometimes it is advantageous to combine these two very different models into an ARMA(p,q) process:

$$x(t) = a_0 + \sum_{i=1}^{p} a_i x(t-i) - \sum_{i=1}^{q} b_i \epsilon(t-i) + \epsilon(t)\,. \tag{A.6}$$

Finally, the *return* of a series P_t at time t is defined as $r_t = \log(P_{t+1}/P_t)$. In the context of modelling heteroscedasticity, an important topic in this report, we often refer to return series.

A2 Heteroscedasticity

Imagine measuring the position of a rocket as it leaves the earth. As the rocket gets further away we are less certain about where it is, and so the variance increases on our measurements. These measurements are an example of a *heteroscedastic* sequence – a sequence of random variables with different variances. Here are some more examples:

- As we get nearer to an election, the results from a polling station become less variable.
- As one's income increases, there is an increasing variability on money spent on a given meal. A rich person may eat at a fancy restaurant, and then later eat fast food. On the other hand, a poor person more consistently eats inexpensive food.

- As a projectile approaches, we are more sure about its position.
- As a man ages from birth, there is fluctuation in our certainty about his health, height, weight, strength, income, and so on. For example, most teenagers earn minimum wage, most elderly are retired and earn very little, and middle-aged adults have highly variable incomes.

We will see examples of heteroscedasticity throughout this dissertation. For example, in chapter 3 we model input dependent signal and noise covariances between multiple responses.

Reasoning optimally in the face of uncertainty is central to statistics. Ignoring heteroscedasticity (input dependent noise or variance) will cause one to make faulty inferences when testing statistical hypotheses. Sometimes it is *just* the uncertainty of a process that changes with time. The log returns on big equity indices like NASDAQ, the S&P 500, the FTSE, and on foreign currency exchanges, are often assumed to have a zero mean, but a time changing variance (Tsay, 2002). Present developments address the need to predict uncertainty in financial markets, somewhat like the theory of thermodynamics initially developed to improve the efficiency of steam engines. But heteroscedasticity is relevant beyond finance, much like thermodynamics is relevant beyond steam engines!

The goal of these models is to predict the latent variance (or covariance matrix in higher dimensions) drawn with observations. For example, suppose we have data points sampled from $\mathcal{N}(0, t^4)$, with variance growing with time t, as shown in Figure A.1. A model of heteroscedasticity should take the information in panel a), and then produce the picture in panel b). It does not fit the observations themselves.

Until Robert Engle developed ARCH (Engle, 1982), for which he won the 2003 Nobel Prize in economics, models of heteroscedasticity were exceptionally limited. The standard model was $y_t = \epsilon_t x_{t-1}$, where y_t is the observation at time t, ϵ_t is a random variable with $\mathrm{Var}(\epsilon_t) = \sigma^2$, and x_t is an exogenous variable. The variance of y_t is then $\sigma^2 x_{t-1}^2$, and so depends solely on the exogenous variable x_t. This method requires one to understand the cause of changing variance. Granger and Andersen (1978) later introduced models that allow the conditional variance

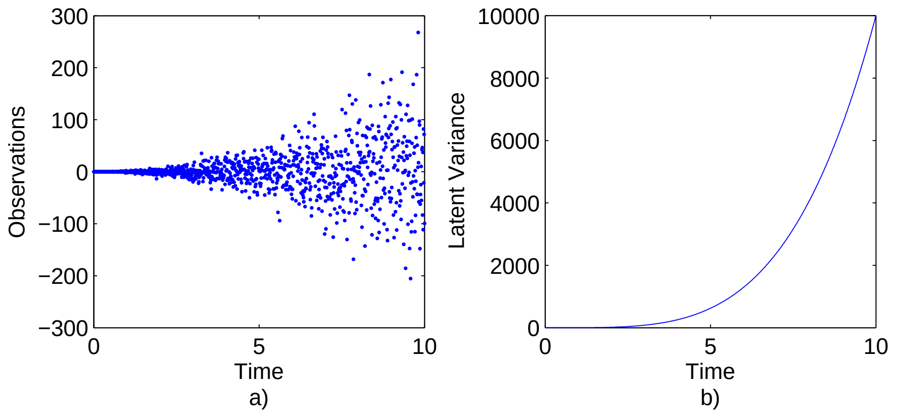


Figure A.1: Determining the latent variance function. In panel a) is a sequence of observations with a time changing variance, and in b) is the latent variance function. A model of heteroscedasticity should take the information in a) and reconstruct the information in b).

to depend on past observations. A simple example is $y_t = \epsilon_t y_{t-1}$, where the conditional variance is $\text{Var}(y_t|y_{t-1}) = \sigma^2 y_{t-1}^2$. However, the unconditional variance is zero or infinity.

We now introduce ARCH (Engle, 1982), and its generalization GARCH (Bollerslev, 1986). GARCH is arguably unsurpassed for predicting the volatility (standard deviation) of returns on equity indices and currency exchanges (Brownlees et al., 2009; Hansen and Lunde, 2005; Poon and Granger, 2005). We then discuss multivariate GARCH, and other models of heteroscedasticity. Consider Figure A.2, which shows the daily returns on NASDAQ, the DOW Jones Composite, and the FTSE, from January 2009 to February 2010. There is a visible relationship between these markets: it is even difficult to tell that these are different plots! Accounting for this relationship will improve volatility predictions for either individual market; in other words, multivariate models can be better than univariate ones at predicting univariate volatility. For a review of multivariate models, see Silvennoinen and Teräsvirta (2009) and Asai et al. (2006).

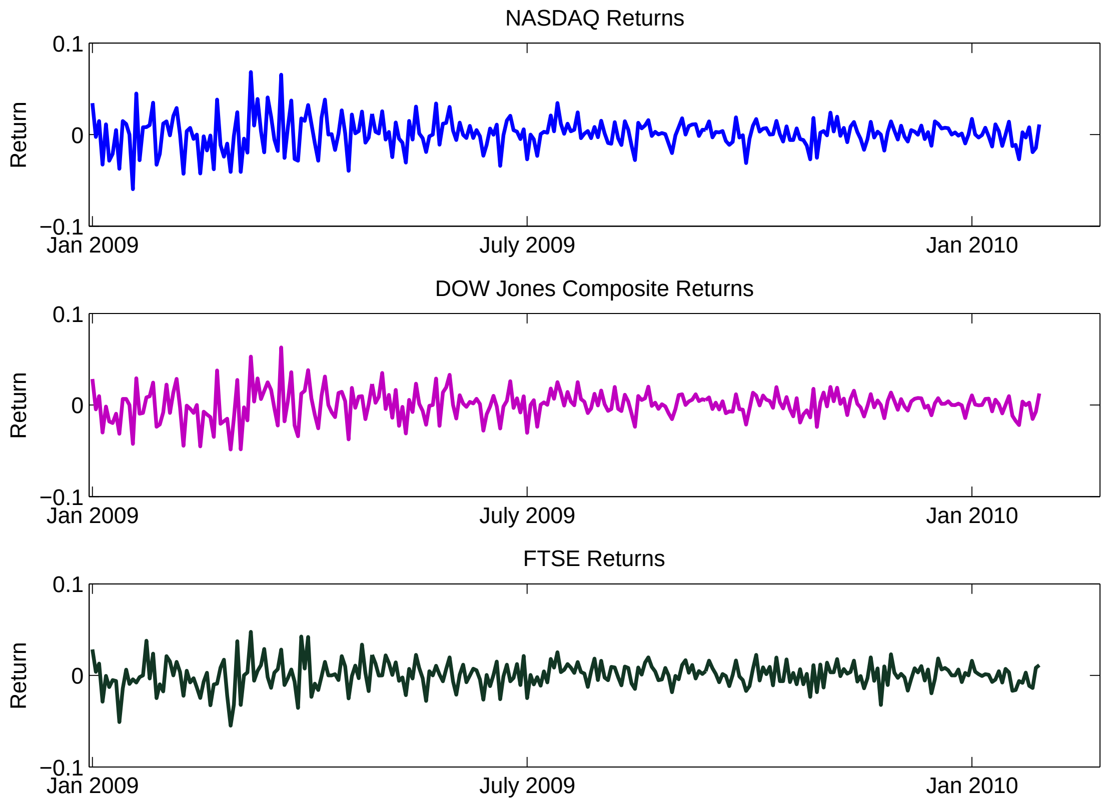


Figure A.2: Visible co-movement between daily returns on NASDAQ, the DOW Jones Composite, and the FTSE.

A22 Univariate ARCH and GARCH

In financial markets, if a return y_t deviates significantly from the mean, the next return will probably do the same. Likewise with small deviations. This is called *volatility clustering*. Since $\mathbb{E}[y_t^2] = \sigma_t^2$, ARCH captures this behaviour by letting the variance at time t deterministically depend on the previous squared returns. In the simplest ARCH model, ARCH(1), the variance depends only on the squared return at the last time-step:

$$y_t = \sigma_t \epsilon_t \tag{A.7}$$

$$\sigma_t^2 = a_0 + a_1 y_{t-1}^2 \tag{A.8}$$

$$\epsilon_t \sim \mathcal{N}(0,1). \tag{A.9}$$

Let Ψ_t represent the information set available at time t. The conditional mean $\mathbb{E}[y_t|\Psi_{t-1}]$ is zero, and the conditional variance $\mathrm{Var}(y_t|\Psi_{t-1})$ is $a_0 + a_1 y_{t-1}^2$. Further, there are no serial correlations in the y_t series, and the unconditional mean is zero:

$$\mathrm{cov}(y_{t+h}, y_t) = \mathbb{E}[y_t y_{t+h}] = \mathbb{E}[\mathbb{E}[y_t y_{t+h}|\Psi_{t+h-1}]] = \mathbb{E}[y_t \mathbb{E}[y_{t+h}|\Psi_{t+h-1}]] = 0, \tag{A.10}$$

$$\mathbb{E}[y_t] = \mathbb{E}[\mathbb{E}[y_t|\Psi_{t-1}]] = \mathbb{E}[\sigma_t \mathbb{E}[\epsilon_t]] = 0. \tag{A.11}$$

It follows that the unconditional variance is constant:

$$\mathrm{Var}(y_t) = \mathbb{E}[y_t^2] = \mathbb{E}[\mathbb{E}[a_0 + a_1 y_{t-1}^2]] = a_0 + a_1 \mathbb{E}[y_{t-1}^2] = a_0 + a_1 \mathrm{Var}(y_t)$$
$$\therefore \mathrm{Var}(y_t) = \frac{a_0}{1 - a_1}. \tag{A.12}$$

Therefore y_t is a white noise series. And though the returns themselves are serially uncorrelated, the squared returns are correlated. In fact, we can rewrite the ARCH(1) series as

$$y_t^2 = a_0 + a_1 y_{t-1}^2 + \sigma_t^2(\epsilon_t^2 - 1)\,, \tag{A.13}$$

which is a non-Gaussian AR(1) model for the squared returns y_t^2 (an unbiased estimator of σ_t^2), hence the name autoregressive conditional heteroscedasticity (ARCH).

From (A.12), a_0 must be greater than zero. Also, the unconditional kurtosis,

$$\kappa = \frac{\mathbb{E}[y_t^4]}{\mathbb{E}[y_t^2]^2} = 3\frac{1-a_1^2}{1-3a_1^2}\,, \tag{A.14}$$

is greater than or equal to three. This means that the marginal distribution of the returns has heavier tails than the normal distribution, even though the returns are conditionally normal; this is good since the actual returns on equity indices and currency exchanges are thought to have heavy tails. However, for the fourth moment $\mathbb{E}[y_t^4]$ to be positive, $0 \leq a_1 \leq 1/3$, which is quite restrictive. The parameters a_0, a_1 can be learned by maximizing the likelihood,

$$p(y_n, y_{n-1}, \ldots, y_2 | y_1, a_0, a_1) = \prod_{t=2}^{n} p(y_t|y_{t-1})\,, \tag{A.15}$$

subject to constraints.

The general ARCH(p) model is

$$y_t = \sigma_t \epsilon_t\,, \tag{A.16}$$

$$\sigma_t^2 = a_0 + \sum_{i=1}^{p} a_i y_{t-i}^2\,, \tag{A.17}$$

$$\epsilon_t \sim \mathcal{N}(0,1)\,, \tag{A.18}$$

and can be rewritten as

$$y_t^2 = a_0 + a_1 y_{t-1}^2 + \cdots + a_p y_{t-p}^2 + \sigma_t^2(\epsilon_t^2 - 1)\,. \tag{A.19}$$

To test whether ARCH is an appropriate model, one can determine the empirical autocorrelations for y_t and y_t^2: the return series should be uncorrelated, unlike the squared return series. To then determine the appropriate order, one can use the PACF. Unfortunately, a high order is often necessary for good predictions – e.g. ARCH(9) for the S&P 500 (Tsay, 2002). To address this issue, Bollerslev (1986) developed GARCH, *generalised autoregressive conditional heteroscedasticity.* The GARCH(p,q) process is given by

$$y_t = \sigma_t \epsilon_t\,, \tag{A.20}$$

$$\sigma_t^2 = a_0 + \sum_{i=1}^{p} a_i y_{t-i}^2 + \sum_{j=1}^{q} b_j \sigma_{t-j}^2\,, \tag{A.21}$$

$$\epsilon_t \sim \mathcal{N}(0,1)\,. \tag{A.22}$$

We can rewrite this as

$$y_t^2 = a_0 + \sum_{i=1}^{\max(p,q)} (a_i + b_i) y_{t-i}^2 - \sum_{j=1}^{q} b_j \eta_{t-j} + \eta_t \,, \tag{A.23}$$

$$\eta_t = y_t^2 - \sigma_t^2 \,. \tag{A.24}$$

Since $\mathbb{E}[\eta_t] = 0$ and $\mathrm{cov}(\eta_t, \eta_{t-j}) = 0$, η_t is a *martingale difference series*, and so (A.23) is a valid ARMA(max(p,q),q) form for y_t^2. So ARCH applies an autoregressive model to y_t^2, and GARCH generalizes this to an ARMA model. Analysis of GARCH is similar to ARCH, and the two models share many properties: y_t is a weakly stationary series with a zero unconditional mean and constant unconditional variance, y_t is serially uncorrelated but y_t^2 isn't, and the kurtosis of y_t is greater than three. It is more difficult to determine the order of a GARCH model in a principled way, but often GARCH(1,1) is appropriate, which makes it popular over the high order ARCH models needed to make comparably accurate predictions. GARCH is usually trained using a constrained maximum likelihood, where an initial σ_0 is somehow specified. ARCH and GARCH share many strengths and weaknesses. They are strong in that they are both simple models, and empirically, they are unsurpassed at predicting volatility in financial markets like the S&P 500. But here are some weaknesses (Tsay, 2002):

- The parameters give no insight into the underlying source of volatility.
- Restrictions are very tight on parameters, e.g. for $\mathbb{E}[y_t^4] > 0$.
- Both often overpredict volatility, as they respond slowly to large isolated returns.
- Positive and negative returns have the same effect on volatility predictions.

It's easy to modify GARCH. For example, the returns y_t could have a different conditional distribution; t-GARCH is a popular variant where the *innovation* ϵ_t has a Student-t distribution. In this case the conditional and unconditional returns have heavier tails. Further, we can remove the undesirable symmetry from GARCH by adding separate terms for positive and negative returns, $a_+ y_+^2$ and $a_- y_-^2$. Nelson

(1991) removes the symmetry with the exponential GARCH (EGARCH) model, which has the innovation (noise model)

$$g(\epsilon_t) = (\theta + \gamma)\epsilon_t - \gamma\mathbb{E}[|\epsilon_t|] \quad \text{if} \quad \epsilon_t \geq 0\,, \tag{A.25}$$

$$g(\epsilon_t) = (\theta - \gamma)\epsilon_t - \gamma\mathbb{E}[|\epsilon_t|] \quad \text{if} \quad \epsilon_t < 0\,, \tag{A.26}$$

where θ and γ are real constants, and, for example, $\epsilon_t \sim \mathcal{N}(0,1)$. A different variant, the GARCH-M model, adds a deterministic volatility term to the expression for the returns,

$$y_t = \mu + c\sigma_t^2 + \sigma_t\epsilon_t\,, \tag{A.27}$$

$$\sigma_t^2 = a_0 + a_1 y_{t-1}^2 + b_1\sigma_{t-1}^2\,, \tag{A.28}$$

in addition to the usual $\sigma_t\epsilon_t$ term. Here $\epsilon_t \sim \mathcal{N}(0,1)$ and μ and c are constants; c is called the *risk premium* parameter. With this new term, the returns y_t are serially correlated. So when historical stock returns have serial correlations, there is said to be *risk premium*. This is common with returns on securities (Tsay, 2002).

Although there are many variations, a simple GARCH(1,1) model is often unsurpassed (Brownlees et al., 2009; Hansen and Lunde, 2005).

A22 Multivariate GARCH

The simplicity of univariate GARCH is arguably its most appealing feature. Unfortunately, with multivariate GARCH (MGARCH) models, the number of parameters increases rapidly with increases in dimension. These parameters become difficult to interpret and estimate. It is also challenging to ensure that the conditional covariance matrix is positive definite. Because of these difficulties, often unrealistic assumptions are made for tractability. In this brief review of MGARCH, we follow Silvennoinen and Teräsvirta (2009) and Tsay (2002).

Let $\boldsymbol{y}_t$ be a D dimensional vector stochastic process, and $\mathbb{E}[\boldsymbol{y}_t] = \mathbf{0}$. Further, let Ψ_t be all information available up to and including time t, and let $\boldsymbol{\eta}_t$ be an iid vector white noise process with $\mathbb{E}[\boldsymbol{\eta}_t\boldsymbol{\eta}_t^\top] = I$. Then, in the general MGARCH framework,

$$\boldsymbol{y}_t = \Sigma_t^{1/2}\boldsymbol{\eta}_t\,, \tag{A.29}$$

where Σ_t is the covariance matrix of $\boldsymbol{y}_t$ conditioned on Ψ_{t-1}.

The first MGARCH model, the VEC model of Bollerslev et al. (1988), specifies Σ_t as

$$\text{vech}(\Sigma_t) = \boldsymbol{a}_0 + \sum_{i=1}^{q} A_i \text{vech}(\boldsymbol{y}_{t-i}\boldsymbol{y}_{t-i}^\top) + \sum_{j=1}^{p} B_j \text{vech}(\Sigma_{t-j})\,. \tag{A.30}$$

A_i and B_j are $D(D+1)/2 \times D(D+1)/2$ matrices of parameters, and $\boldsymbol{a}_0$ is a $D(D+1)/2 \times 1$ vector of parameters. The vech operator stacks the columns of the lower triangular part of a $D \times D$ matrix into a vector of size $D(D+1)/2 \times 1$. For example, $\text{vech}(\Sigma) = (\Sigma_{11}, \Sigma_{21}, \dots, \Sigma_{D1}, \Sigma_{22}, \dots, \Sigma_{D2}, \dots, \Sigma_{DD})^\top$. This model is general, but difficult to use. There are $(p+q)(D(D+1)/2)^2 + D(D+1)/2$ parameters! These parameters are hard to interpret, and there are no conditions under which Σ_t is positive definite for all t. Gouriéroux (1997) discusses the challenging (and sometimes impossible) problem of keeping Σ_t positive definite. Training is done by a constrained maximum likelihood, where the log likelihood is given by

$$\mathcal{L} = \boldsymbol{a}_0 - \frac{1}{2}\sum_{t=1}^{N} \log|\Sigma_t| - \frac{1}{2}\sum_{t=1}^{N} \boldsymbol{y}_t^\top \Sigma_t^{-1} \boldsymbol{y}_t\,, \tag{A.31}$$

supposing that $\boldsymbol{\eta}_t \sim \mathcal{N}(0, I)$, and that there are N training points.

Subsequent efforts have led to simpler but less general models. We can let A_j and B_j be diagonal matrices. This model has notably fewer (though still $(p+q+1)D(D+1)/2$) parameters, and there are conditions under which Σ_t is positive definite for all t (Engle et al., 1994). But now there are no interactions between the different conditional variances and covariances.

Factor models are another popular simplification. The idea is to find the few factors that explain most of the variability in the returns $\boldsymbol{y}_t$. Often these factors are chosen using principle component analysis (PCA) (Hotelling, 1933; Jolliffe, 2002; Pearson, 1901). This reduces the dimension of the modelling problem. Engle et al. (1990) define a factor volatility model. As described by Silvennoinen and Teräsvirta (2009), they assume Σ_t is generated from $K < D$ factors $f_{k,t}$:

$$\Sigma_t = A + \sum_{k=1}^{K} \boldsymbol{w}_k \boldsymbol{w}_k^\top f_{k,t}\,, \tag{A.32}$$

where A is a $D \times D$ positive semidefinite matrix, and each $\boldsymbol{w}_k$ is a $D \times 1$ vector of factor weights. These factors have a GARCH(1,1) form

$$f_{k,t} = c_k + a_k(\boldsymbol{v}^\top \boldsymbol{y}_{t-1})^2 + b_k f_{k,t-1}\,, \tag{A.33}$$

where c_k, a_k, b_k are scalar constants, and $\boldsymbol{v}$ is a $D \times 1$ vector of weights. For a detailed review of multivariate GARCH models, see Silvennoinen and Teräsvirta (2009).

A22 Stochastic Volatility Models

ARCH and GARCH let the volatility be a deterministic function of the past. A popular alternative class of models, *stochastic volatility* models, let the volatility be a stochastic process. For example,

$$y_t = \sigma_t \epsilon_t \tag{A.34}$$
$$\sigma_t = \exp(M_t) \tag{A.35}$$
$$\epsilon_t \sim \mathcal{N}(0,1), \tag{A.36}$$

where M_t is a stochastic model, like an AR process. For example, Wilson and Ghahramani (2010a) develop a stochastic volatility model, Gaussian Process Copula Volatility (GCPV), where σ_t is transformed so that it is best modelled by a Gaussian Process (chapter 2). An early stochastic volatility model by Harvey et al. (1994) has

$$y_t = \sigma_t \epsilon_t\,, \tag{A.37}$$
$$\log(\sigma_t^2) = a_0 + a_1 \log(\sigma_{t-1}^2) + w_t\,, \tag{A.38}$$

where w_t is Gaussian white noise with variance σ_w^2. The log variance follows an AR(1) process. Letting $g_t = \log y_t^2$ we can rewrite (A.37) as

$$g_t = \log(\sigma_t^2) + \log(\epsilon_t^2). \tag{A.39}$$

Together (A.38) and (A.39) would form a state-space model if ϵ_t^2 had a log-normal distribution. Usually, though, $\epsilon_t \sim \mathcal{N}(0,1)$.

There are multivariate extensions to stochastic volatility models. They suffer from many of the same problems as multivariate GARCH – the number of parameters scales badly with dimension, so many unrealistic assumptions are made, and it's difficult to keep the covariance matrices positive definite. For a review, see Asai et al. (2006).

Appendix B

Generalised Wishart Processes

We discuss properties of the GWP construction we introduced in section 3.8, including support, moments, autocorrelations, and stationarity. We also interpret its free parameters. We then introduce several alternate GWP constructions, with benefits in expressivity and efficiency. As our discussion progresses the generalised Wishart process class of models will become increasingly clear. Inference for the GWP is discussed in Wilson and Ghahramani (2010b, 2011).

B1 Properties

Since the marginal distributions are Wishart – that is, $\Sigma(x)$ has a Wishart distribution for any $x \in \mathcal{X}$ – we know that at every x there is support for every positive definite covariance matrix. We also know

$$\mathbb{E}[\Sigma(x)] = \nu V \,, \tag{B.1}$$

$$\mathrm{Var}[\Sigma_{ij}(x)] = \nu(V_{ij}^2 + V_{ii}V_{jj}) \,, \tag{B.2}$$

and we can find simple expressions for the covariances between entries of $\Sigma(x)$ and $\Sigma(x')$.

Theorem B1.1. *Assuming L is diagonal, and that Gaussian process u_{id} in Eq.* (3.20) *has kernel k_d for each dimension ($d = 1, \dots, p$),*[1] *then for $i \neq j \neq l \neq s$,*

$$cov[\Sigma_{ii}(x), \Sigma_{ii}(x')] = 2\nu L_{ii}^4 k_i(x,x')^2 \tag{B.3}$$

$$cov[\Sigma_{ij}(x), \Sigma_{ji}(x')] = cov(\Sigma_{ij}(x), \Sigma_{ij}(x')) = \nu L_{ii}^2 L_{jj}^2 k_i(x,x') k_j(x,x') \tag{B.4}$$

$$cov[\Sigma_{ij}(x), \Sigma_{ls}(x')] = 0\,. \tag{B.5}$$

Proof B1.1. *We can rewrite $\Sigma(x)$ in* (3.22) *as a sum of ν i.i.d. matrices A_j: $\Sigma(x) = \sum_{j=1}^{\nu} A_j(x)$. Let $A = A_1$. Then $cov[\Sigma(x), \Sigma(x')] = \nu cov[A(x), A(x')]$. Further, $cov[A_{ii}(x), A_{ii}(x')] = L_{ii}^4 cov[u_{ii}(x)^2, u_{ii}(x')^2]$. Since $u_{ii}(x)$ and $u_{ii}(x')$ are each $\mathcal{N}(0,1)$, and they are jointly Gaussian, we can write*

$$u_{ii}(x') = k_i(x,x') u_{ii}(x) + \sqrt{1 - k_i(x,x')^2}\epsilon \tag{B.6}$$

where ϵ is $\mathcal{N}(0,1)$ and independent of both $u_{ii}(x)$ and $u_{ii}(x')$. Then, abbreviating $k_i(x,x')$ as k_i,

$$u_{ii}(x')^2 = k_i^2 u_{ii}(x)^2 + 2k_i\sqrt{1-k_i^2}\epsilon u_{ii}(x) + (1-k_i^2)\epsilon^2\,. \tag{B.7}$$

Therefore

$$\mathbb{E}[u_{ii}(x)^2 u_{ii}(x')^2] = \mathbb{E}[k_i^2 u_{ii}(x)^4 + 2k_i\sqrt{1-k_i^2}\epsilon u_{ii}(x)^2 + (1-k_i^2)\epsilon^2 u_{ii}(x)^2]\,, \tag{B.8}$$

$$= 3k_i^2 + (1-k_i^2) = 2k_i^2 + 1\,, \tag{B.9}$$

and so

$$cov[u_{ii}(x)^2, u_{ii}(x')^2] = \mathbb{E}[u_{ii}(x)^2 u_{ii}(x')^2] - \mathbb{E}[u_{ii}(x)^2]\mathbb{E}[u_{ii}(x')^2] \tag{B.10}$$

$$= 2k_i^2 + 1 - 1 = 2k_i^2\,. \tag{B.11}$$

We conclude

$$cov[\Sigma_{ii}(x), \Sigma_{ii}(x')] = 2\nu L_{ii}^4 k_i(x,x')^2\,. \tag{B.12}$$

The derivations of equations (B.4) *and* (B.5) *are similar.*

Corollary B1.1. *Since $\mathbb{E}[\Sigma(x)] = \nu V$ = constant, and the covariances are proportional to the kernel function(s), the Generalised Wishart Process is weakly stationary if the kernel function(s) are stationary – that is, if $k(x,x') = k(x+a, x'+a)$ for all feasible constants a. This holds, for example, if $k(x,x')$ is a function of $||x - x'||$.*

[1]The idea of a new kernel function for each dimension is discussed further in section B2.

Corollary B1.2. *If $k(x, x') \to 0$ as $||x - x'|| \to \infty$ then*

$$\lim_{||x-x'||\to\infty} cov[\Sigma_{ij}(x), \Sigma_{ls}(x')] = 0\,, \quad \forall\; i, j, l, s \in \{1, \dots, p\}\,. \tag{B.13}$$

We can also derive expressions for the covariances when L is full rank, and for when there are different kernel functions for each degree of freedom. But the conclusions are qualitatively the same: the covariances are directly proportional to the kernel function(s), and Corollaries B1.1 and B1.2 remain unchanged.

To find the autocorrelations, one can normalise the covariances in (B.3)-(B.5). Since $k(x, x) = 1$ for all $x \in \mathcal{X}$, the autocorrelations take an especially simple form. For example, assuming that the kernel is stationary (that is $k(a + x, a)$ is the same for all $a \in \mathcal{X}$), then the *lag-x* autocorrelation function (ACF) is

$$\mathrm{ACF}_{ij}(x) = \mathrm{corr}(\Sigma_{ij}(0), \Sigma_{ij}(x)) = \begin{cases} k_i(0, x)^2 & \text{if } i = j\,, \\ k_i(0, x)k_j(0, x) & \text{if } i \neq j\,. \end{cases} \tag{B.14}$$

This again makes clear the direct relationship between the kernel $k(x, x')$ (and its parameters, like *length-scale*) and the dynamics of the matrix $\Sigma(x)$. In fact one way to set the values of parameters like length-scale is to use the empirical autocorrelation function.

We conclude this section with a property that has a special meaning in finance.
Theorem B1.2. *If $\Sigma(x) \sim \mathcal{GWP}(V, \nu, k)$ then $A^\top\Sigma(x)A \sim \mathcal{GWP}(A^\top VA, \nu, k)$.*
Proof B1.2. *From* (3.22),

$$A^\top\Sigma(x)A = \sum_{i=1}^{\nu} A^\top L\hat{\boldsymbol{u}}_i(x)\hat{\boldsymbol{u}}_i^\top(x)L^\top A = \sum_{i=1}^{\nu} \boldsymbol{w}_i(x)\boldsymbol{w}_i^\top(x)\,, \tag{B.15}$$

where $\boldsymbol{w}_i \sim \mathcal{N}(0, A^\top VA)$.

Suppose we have p assets with returns $\boldsymbol{r}(t)$, and a $p \times p$ matrix A, where each column contains the proportion invested in each asset for a given portfolio. The returns on the portfolios are $A^\top\boldsymbol{r}(t)$ and the volatility is $A^\top\Sigma(t)A$. Therefore if asset return volatility follows a Wishart process $\mathcal{GWP}(\nu, V, k)$ then the volatility of the p portfolios also follows a Wishart process. This means that, unlike MGARCH models, the Wishart process is invariant to portfolio allocation (Gouriéroux et al., 2009).

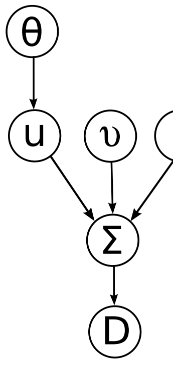


Figure B.1: Graphical model of the generalised Wishart process. $\boldsymbol{u}$ is a vector of all GP function values, $\boldsymbol{\theta}$ are GP hyperparameters, L is the lower Cholesky decomposition of the scale matrix ($LL^\top = V$), ν are the degrees of freedom, and Σ is the covariance matrix.

B2 Interpretation of Model Parameters

In Figure 3.4 we showed a draw of $\Sigma(x)$ evolving with x (e.g. time). Considering just one of these time steps or spatial locations, Figure B.1 shows the conditional dependency relationships for all parameters in the construction of section 3.8.

The kernel function(s) k, and the free parameters $L, \boldsymbol{\theta}, \nu$ have clear and meaningful roles. Ultimately we wish to infer the posterior $p(\Sigma(x)|\mathcal{D})$ for all x, and in equations (B.1) and (B.2) we see that L sets the prior expectation and variance for $\Sigma(x)$ at every x (recall that $V = LL^\top$). One can take $L = \text{chol}(\tilde{\Sigma}/\nu)$, where $\tilde{\Sigma}$ is the empirical covariance matrix for $\boldsymbol{y}$ at all observed x (e.g. not at the locations x we wish to forecast). Sampling from the distribution $p(L|\mathcal{D})$ is another option discussed further in Wilson and Ghahramani (2011).

There is some redundancy between ν and L, like in equations (B.1) and (B.2) for the mean and variance of $\Sigma(x)$; for any ν we can scale L so that $\mathbb{E}[\Sigma(x)] = \nu LL^\top$ is some constant matrix A, by letting $L = A^{1/2}/\sqrt{\nu}$. However, suppose we do scale L in this way as ν grows. From (B.2), $\text{Var}[\Sigma_{ij}(x)] = \frac{1}{\nu}(A_{ii} + A_{ii}A_{jj})$ which $\to 0$ as $\nu \to \infty$. Equivalently, as ν increases, samples of $\Sigma(x)$ become relatively close to one another. So ν controls how broad our prior is on $\Sigma(x)$ at each x. One can sample from the posterior distribution over ν using slice sampling. This posterior distribution tells us how much we ought to trust our prior over $\Sigma(x)$. Alternatively, if we have little prior knowledge of $\Sigma(x)$, then we might simply fix ν to a small value. The smallest we can make ν is $p+1$ in order for $\Sigma(x)$ to stay positive definite (however, one can use a factor representation and add to the diagonal of the matrix, as in Eq. (3.19)). This is a drawback of the Wishart distribution. In section B4 we introduce a t-GWP, which is more robust to the limitation $\nu > p$.

The kernel function $k(x, x')$ is also of great importance[1]. As we saw in the previous section the kernel controls the autocorrelations. The kernel also determines whether or not the dynamics of $\Sigma(x)$ are Markovian, periodic, smooth, etc. The parameters of the kernel function $\boldsymbol{\theta}$ tell us about the underlying source of volatility. They address questions like: Is there periodicity? If so, what is the period? How much past data is needed to make a good forecast? And how quickly do the autocorrelations change over time? The last two questions are answered by the *length-scale* parameter. Since the answers will sometimes be different for different univariate time series, we may want different length-scales for each dimension of $\boldsymbol{y}$ – meaning a new kernel function k_d for each dimension ($d = 1, \dots, p$). This can be easily handled in the model construction: we just let the Gaussian process $u_{id}(x)$ have kernel k_d. Quite often this is unnecessary, since the dimensions of $\boldsymbol{y}(t)$ will be different time series with similar length-scales, like returns for NASDAQ, S&P 500, DJI, FTSE, etc.

Overall, the free parameters in the GWP have clear interpretations and, when learned from data, provide fundamental insights about what we are modelling.

B3 Alternate Construction for Real Valued Degrees of Freedom

We now introduce a new Generalised Wishart Process construction which is more expressive than the previous construction, because it has real valued degrees of freedom ν.[2] Moreover, the number of necessary stochastic processes is independent

[1]Covariance kernels are described in detail in section 2.4.

[2]We thank John P. Cunningham for helpful conversations and for bringing the Bartlett (1933) construction of the Wishart distribution to our attention.

of ν (in the previous construction there were $p\nu$ Gaussian processes). And it is easier to estimate ν in this new formulation, where standard sampling techniques can be applied, without the need for reversible jump MCMC, for example.

Like before, we start with a construction of the standard Wishart distribution, and then show how to create a process from it. Let

$$A = \begin{pmatrix} \sqrt{m_{11}} & 0 & 0 & \cdots & 0 \\ u_{21} & \sqrt{m_{22}} & 0 & \cdots & 0 \\ u_{31} & u_{32} & \sqrt{m_{33}} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{p1} & u_{p2} & u_{p3} & \cdots & \sqrt{m_{pp}} \end{pmatrix}, \tag{B.16}$$

where $m_{ii} \sim \chi^2(\nu - i + 1)$ and $u_{ij} \sim \mathcal{N}(0,1)$. Then $LAA^\top L^\top \sim W_p(\nu, LL^\top)$ (Bartlett, 1933). We can allow for real valued degrees of freedom by instead letting $m_{ii} \sim \Gamma(\frac{\nu-i+1}{2}, 2)$. To turn this into a Wishart process, we replace A by $A(x)$, where

$$A(x) = \begin{pmatrix} \sqrt{m_{11}(x)} & 0 & 0 & \cdots & 0 \\ u_{21}(x) & \sqrt{m_{22}(x)} & 0 & \cdots & 0 \\ u_{31}(x) & u_{32}(x) & \sqrt{m_{33}(x)} & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ u_{p1}(x) & u_{p2}(x) & u_{p3}(x) & \cdots & \sqrt{m_{pp}(x)} \end{pmatrix}. \tag{B.17}$$

The $u_{ij}(x) \sim \mathcal{GP}(0,k)$, with $k(x,x) = 1$, and $m_{ii}(x)$ are *Gaussian Copula Processes* [1] (Wilson and Ghahramani, 2010a) that are marginally Gamma$(\frac{\nu-i+1}{2}, 2)$ at every x. Specifically, for each x, $m_{ii}(x) = G_i^{-1}\Phi(u_{ii}(x))$, where $u_{ii} \sim \mathcal{GP}(0,k)$ with $k(x,x) = 1$, Φ is the standard univariate Gaussian cdf, and G_i^{-1} is the inverse cdf of the Gamma$(\frac{\nu-i+1}{2}, 2)$ distribution. We then construct $\Sigma(x)$ as

$$\Sigma(x) = LA(x)A(x)^\top L^\top, \tag{B.18}$$

which has Wishart marginals $W_p(\nu, LL^\top)$, with real valued degrees of freedom ν, at every x.

[1] A Gaussian Copula Process has an underlying Gaussian process dependency structure, but arbitrary marginal distributions. A copula process is used to specify a meaningful dependency structure between arbitrarily many random variables with arbitrary marginal distributions. See Wilson and Ghahramani (2010a) for more on copula processes.

Notice that the number of Gaussian processes is $p(p+1)/2$ compared to the $p\nu \geq p(p+1)$ in the previous construction. There are at least half as many GPs, and the number of GPs is independent of the degrees of freedom. So this construction can be more efficient than the construction in section 3.8; however, for small ν (e.g. $\nu < 5$) this is typically not the case, because the inverse Gamma cdf requires some computation.

B4 t-Wishart Process Construction

We now develop a GWP construction with heavy tails. We again start by constructing a matrix variate distribution, and then formulate a process from it. This time, however, the matrix variate distribution belongs to a more general class of distributions which contains the standard Wishart distribution as a special case. It is constructed from special multivariate Student-t random variables.[1]

A p-variate Student-t random variable $\boldsymbol{z}$ with m degrees of freedom typically has the pdf

$$p(\boldsymbol{z}) = \frac{\Gamma((m+p)/2)}{(\pi m)^{p/2}\Gamma(m/2)|V|^{1/2}}\left[1+\frac{1}{m}(\boldsymbol{z}-\boldsymbol{\mu})^\top V^{-1}(\boldsymbol{z}-\boldsymbol{\mu})\right]^{-(m+p)/2}, \tag{B.19}$$

where $V = \mathrm{cov}[\boldsymbol{z}]$. The joint pdf of ν independent such variables is $p(\boldsymbol{z}_1, \dots, \boldsymbol{z}_\nu) = p(\boldsymbol{z}_1)\cdots p(\boldsymbol{z}_\nu)$.

For dependent but uncorrelated multivariate t variables, Joarder and Ahmed (1996) proposed

$$p(\boldsymbol{z}_1, \dots, \boldsymbol{z}_\nu) = \frac{\Gamma((m+p)/2)}{(\pi^\nu m)^{p/2}\Gamma(m/2)|V|^{1/2}}\left[1+\frac{1}{m}(\boldsymbol{z}-\boldsymbol{\mu})^\top V^{-1}(\boldsymbol{z}-\boldsymbol{\mu})\right]^{-(m+\nu p)/2}. \tag{B.20}$$

The marginals $\boldsymbol{z}_i$ are p-variate t as in (B.19) (Kotz and Nadarajah, 2004). Joarder and Ali (1997) showed that (B.20) can be expressed as a scale mixture of multi-

[1]Kotz and Nadarajah (2004) contains a thorough discussion of multivariate t distributions.

variate Gaussians,

$$p(\boldsymbol{z}_1, \ldots, \boldsymbol{z}_\nu) = \int_0^\infty \frac{|\tau^2 V|^{-\nu/2}}{2\pi^{\nu p/2}} \exp\left[-\frac{1}{2}\sum_{i=1}^{\nu}(\boldsymbol{z}_i - \boldsymbol{\mu})^\top (\tau^2 V)^{-1}(\boldsymbol{z}_i - \boldsymbol{\mu})\right] h(\tau) d\tau\,, \tag{B.21}$$

where τ is inverse Gamma with pdf

$$h(\tau) = \frac{2\tau^{-(1+m)}}{(2/m)^{m/2}\Gamma(m/2)} \exp[-\frac{m}{2\tau^2}]\,. \tag{B.22}$$

Therefore $\boldsymbol{z}_i|\tau \sim \mathcal{N}(\boldsymbol{\mu}, \tau^2 V)$. Zellner (1976) and Sutradhar and Ali (1986) used (B.20) in financial applications, and Kelejian and Prucha (1985) showed that (B.20) is better at capturing heavy tailed behaviour than the standard independent t model.

In analogy with the construction of the Wishart distribution, we consider the matrix A formed by taking a sum of outer products of zero mean $\boldsymbol{z}_i$ (where $\boldsymbol{z}_i, \ldots, \boldsymbol{z}_\nu$ have the joint pdf in (B.20)):

$$A = \sum_{i=1}^{\nu} \boldsymbol{z}_i \boldsymbol{z}_i^\top\,. \tag{B.23}$$

Sutradhar and Ali (1989) derived the corresponding pdf

$$p(A) = \frac{\Gamma((m+2)/2)}{m^{p/2}\Gamma(m/2)\Gamma_p(\nu/2)} |V|^{-(\nu-1)/2} |A|^{-(\nu-p-2)/2} \left[m + \mathrm{tr}(V^{-1}A)\right]^{-(m+p(\nu-1))/2}\,. \tag{B.24}$$

Γ_p is the p dimensional Gamma function and $m > p+1$. The matrix variate distribution in (B.24) converges to the standard Wishart distribution $\mathcal{W}_p(\nu - 1, V)$ as the degrees of freedom $m \to \infty$. The expected value of A is

$$\mathbb{E}[A] = \frac{(\nu-1)}{1-2/m} V\,, \tag{B.25}$$

and for fixed m, there is more probability mass assigned to positive definite matrices “far away” from this expectation. We say that A is *t-Wishart*, $A \sim \mathcal{TW}(\nu, m, V)$, with degrees of freedom ν and m.

We wish to construct a process over covariance matrices $\Sigma(x)$ with *t-Wishart* marginal distributions at every $x \in \mathcal{X}$. In analogy with (3.22), we start with

$\hat{\boldsymbol{u}}_i(x) = (u_{i1}(x), \dots, u_{ip}(x))^\top$, where $i = 1, \dots, \nu$ and $u_{id}(x)$ are independent Gaussian processes, $u_{id}(x) \sim \mathcal{GP}(0, k)$ with $k(t,t) = 1$. So that we can form a Wishart matrix like (B.23) at every x, we want to transform $\hat{\boldsymbol{u}}_i(x)$ to $\hat{\boldsymbol{z}}_i(x) = (z_{i1}(x), \dots, z_{ip}(x))^\top$. $\hat{\boldsymbol{z}}_i(x)$ must be p-variate t as in (B.19), with mean 0, covariance matrix I, and degrees of freedom m. Further, $\hat{\boldsymbol{z}}_1(x), \dots, \hat{\boldsymbol{z}}_\nu(x)$ must have the joint pdf in (B.20), with mean 0, $\nu p \times \nu p$ covariance matrix I, and m degrees of freedom.

To form $\hat{\boldsymbol{z}}_i(x)$ from $\hat{\boldsymbol{u}}_i(x)$ we rely on the scale mixture representation in (B.21). $\hat{\boldsymbol{z}}_i(x) = \tau(x)\hat{\boldsymbol{u}}_i(x)$, where $\tau(x)$ is an inverse Gamma random variable for every $x \in \mathcal{X}$. We then must specify $\tau(x)$. One possibility is to let $\tau(x) = G^{-1}[\Phi(w(x))]$ for every $x \in \mathcal{X}$, where $w(x) \sim \mathcal{GP}(0, k)$, Φ is a Gaussian cdf, and G is the Gamma cdf. This is another example of a Gaussian copula process (Wilson and Ghahramani, 2010a). Now that we have $\hat{\boldsymbol{z}}_i(x)$ we can form the new process as in Theorem B4.1.

Theorem B4.1. *The process defined by*

$$\Sigma(x) = \sum_{i=1}^{\nu} L\hat{\boldsymbol{z}}_i(x)\hat{\boldsymbol{z}}_i(x)^\top L^\top \tag{B.26}$$

has a t-Wishart marginal distribution $\mathcal{TW}(\nu, m, LL^\top)$ at every $x \in \mathcal{X}$.

Proof B4.1. *This is an extension of* (3.22) *which follows from* (B.23).

Appendix C

Derivations for Spectral Mixture Kernels

A stationary kernel $k(x, x')$ is the inverse Fourier transform of its spectral density $S(s)$,

$$k(\tau) = \int S(s) e^{2\pi i s^\top \tau} ds \,, \tag{C.1}$$

where $\tau = x - x'$. First suppose

$$S(s) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{1}{2\sigma^2}(s - \mu)^2\} \,, \tag{C.2}$$

where s, μ, σ and $\tau = x - x'$ are scalars. Substituting (C.2) into (C.1),

$$k(x,x') = \int \exp(2\pi i s(x-x')) \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(s-\mu)^2) ds \tag{C.3}$$

let $\tau = x - x'$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \int \exp[2\pi i s\tau - \frac{1}{2\sigma^2}(s^2 - 2\mu s + \mu^2)] ds \tag{C.4}$$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \int \exp[-\frac{1}{2\sigma^2}s^2 + (2\pi i\tau + \frac{\mu}{\sigma^2})s - \frac{\mu^2}{2\sigma^2}] ds \tag{C.5}$$

let $a = \frac{1}{2\sigma^2}, b = 2\pi i\tau + \frac{\mu}{\sigma^2}, c = -\frac{\mu^2}{2\sigma^2}$

$$= \frac{1}{\sqrt{2\pi\sigma^2}} \int \exp(-a(s - \frac{b}{2a})^2) \exp(\frac{b^2}{4a} + c) ds \tag{C.6}$$

$$= \exp[(2\pi i\tau + \frac{\mu}{\sigma^2})^2 \frac{\sigma^2}{2} - \frac{\mu^2}{2\sigma^2}] \tag{C.7}$$

$$= \exp[(-4\pi^2\tau^2 + 4\pi i\tau\frac{\mu}{\sigma^2} + \frac{\mu^2}{\sigma^4})\frac{\sigma^2}{2} - \frac{\mu^2}{2\sigma^2}] \tag{C.8}$$

$$= \exp[-2\pi^2(x-x')^2\sigma^2][\cos(2\pi(x-x')\mu) + i\sin(2\pi(x-x')\mu))]\,. \tag{C.9}$$

Noting that the spectral density $S(s)$ must be symmetric about $s = 0$, we let

$$\phi(s\,;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\{-\frac{1}{2\sigma^2}(s-\mu)^2\}, \quad \text{and} \tag{C.10}$$

$$S(s) = [\phi(s) + \phi(-s)]/2\,. \tag{C.11}$$

Closely following the above derivation, substituting (C.11) into (C.1) gives

$$k(\tau) = \exp\{-2\pi^2\tau^2\sigma^2\}\cos(2\pi\tau\mu)\,. \tag{C.12}$$

If $\phi(s)$ is instead a mixture of Q Gaussians on $\mathbb{R}^P$, where the q^{th} component has mean vector $\boldsymbol{\mu}_q = (\mu_q^{(1)}, \dots, \mu_q^{(P)})$ and covariance matrix $V = \text{diag}(v_q^{(1)}, \dots, v_q^{(P)})$, and τ_p is the p^{th} component of the P dimensional vector $\tau = x - x'$, then it similarly follows that

$$k(\tau) = \sum_{q=1}^{Q} w_q \cos(2\pi\boldsymbol{\mu}_q^\top\tau) \prod_{p=1}^{P} \exp\{-2\pi^2\tau_p^2 v_q^{(p)}\}. \tag{C.13}$$

Appendix D

GPatt

D1 Introduction

We provide further detail about the eigendecomposition of kronecker matrices, and the runtime complexity of kronecker matrix vector products. Sections D3 and D4 were largely contributed by Elad Gilboa.

D2 Eigendecomposition of Kronecker Matrices

Assuming a product kernel,

$$k(x_i, x_j) = \prod_{p=1}^{P} k^p(x_i^p, x_j^p)\,, \tag{D.1}$$

and inputs $x \in \mathcal{X}$ on a multidimensional grid, $\mathcal{X} = \mathcal{X}_1 \times \cdots \times \mathcal{X}_P \subset \mathbb{R}^P$, then the covariance matrix K decomposes into a Kronecker product of matrices over each input dimension $K = K^1 \otimes \cdots \otimes K^P$ (Saatchi, 2011). The eigendecomposition of K into $QVQ^\top$ similarly decomposes: $Q = Q^1 \otimes \cdots \otimes Q^P$ and $V = V^1 \otimes \cdots \otimes V^P$. Each covariance matrix K^p in the Kronecker product has entries $K_{ij}^p = k^p(x_i^p, x_j^p)$ and decomposes as $K^p = Q^p V^p Q^{p\top}$. Thus the $N \times N$ covariance matrix K can be stored in $\mathcal{O}(PN^{\frac{2}{P}})$ and decomposed into $QVQ^\top$ in $\mathcal{O}(PN^{\frac{3}{P}})$ operations, for N

datapoints and P input dimensions. [1]

D3 Matrix-vector Product for Kronecker Matrices

We first define a few operators from standard Kronecker literature. Let $\mathbf{B}$ be a matrix of size $p \times q$. The reshape$(\mathbf{B}, r, c)$ operator returns a r-by-c matrix ($rc = pq$) whose elements are taken column-wise from $\mathbf{B}$. The vec$(\cdot)$ operator stacks the matrix columns onto a single vector, vec$(\mathbf{B})$ = reshape$(\mathbf{B}, pq, 1)$, and the vec$^{-1}(\cdot)$ operator is defined as vec$^{-1}($vec$(\mathbf{B})) = \mathbf{B}$. Finally, using the standard Kronecker property $(\mathbf{B} \otimes \mathbf{C})\text{vec}(\mathbf{X}) = \text{vec}(\mathbf{C}\mathbf{X}\mathbf{B}^\top)$, we note that for any N element vector $\mathbf{u} \in \mathbb{R}^N$ we have

$$\mathbf{K}_N \mathbf{u} = \left(\bigotimes_{p=1}^{P} \mathbf{K}^p_{N^{1/P}} \right) \mathbf{u} = \text{vec}\left(\mathbf{K}^P_{N^{1/P}} \mathbf{U} \left(\bigotimes_{p=1}^{P-1} \mathbf{K}^p_{N^{1/P}} \right)^\top \right), \tag{D.2}$$

where $\mathbf{U} = \text{reshape}(\mathbf{u}, N^{1/P}, N^{\frac{P-1}{P}})$, and $\mathbf{K}_N$ is an $N \times N$ Kronecker matrix. With no change to Eq. (D.2) we can introduce the vec$^{-1}($vec$(\cdot))$ operators to get

$$\mathbf{K}_N \mathbf{u} = \text{vec}\left(\left(\text{vec}^{-1}\left(\text{vec}\left(\left(\bigotimes_{p=1}^{P-1} \mathbf{K}^p_{N^{1/P}} \right) \left(\mathbf{K}^P_{N^{1/P}} \mathbf{U} \right)^\top \right) \right) \right)^\top \right). \tag{D.3}$$

The inner component of Eq. (D.3) can be written as

$$\text{vec}\left(\left(\bigotimes_{p=1}^{P-1} \mathbf{K}^p_{N^{1/P}} \right) \left(\mathbf{K}^P_{N^{1/P}} \mathbf{U} \right)^\top \mathbf{I}_{N^{1/P}} \right) = \mathbf{I}_{N^{1/P}} \otimes \left(\bigotimes_{p=1}^{P-1} \mathbf{K}^p_{N^{1/P}} \right) \text{vec}\left(\left(\mathbf{K}^P_{N^{1/P}} \mathbf{U} \right)^\top \right). \tag{D.4}$$

Notice that Eq. (D.4) is in the same form as Eq. (D.2) (Kronecker matrix-vector product). By repeating Eqs. (D.3-D.4) over all P dimensions, and noting that

[1]The total number of datapoints $N = \prod_p |\mathcal{X}_p|$, where $|\mathcal{X}_p|$ is the cardinality of $\mathcal{X}_p$. For clarity of presentation, we assume each $|\mathcal{X}_p|$ has equal cardinality $N^{1/P}$.

$\left(\bigotimes_{p=1}^{P} \mathbf{I}_{N^{1/P}}\right) \mathbf{u} = \mathbf{u}$, we see that the original matrix-vector product can be written as

$$\left(\bigotimes_{p=1}^{P} \mathbf{K}^{p}_{N^{1/P}}\right) \mathbf{u} = \mathrm{vec}\left(\left[\mathbf{K}^{1}_{N^{1/P}}, \ldots \left[\mathbf{K}^{P-1}_{N^{1/P}}, \left[\mathbf{K}^{P}_{N^{1/P}}, \mathbf{U}\right]\right]\right]\right) \tag{D.5}$$

$$\overset{\mathrm{def}}{=} \mathrm{kron\_mvprod}\left(\mathbf{K}^{1}_{N^{1/P}}, \mathbf{K}^{2}_{N^{1/P}}, \ldots, \mathbf{K}^{P}_{N^{1/P}}, \mathbf{u}\right) \tag{D.6}$$

where the bracket notation denotes matrix product, transpose then reshape, i.e.,

$$\left[\mathbf{K}^{p}_{N^{1/P}}, \mathbf{U}\right] = \mathrm{reshape}\left(\left(\mathbf{K}^{p}_{N^{1/P}}\mathbf{U}\right)^{\top}, N^{1/P}, N^{\frac{P-1}{P}}\right). \tag{D.7}$$

Iteratively solving the kron_mvprod operator in Eq. (D.6) requires $PN^{\frac{P+1}{P}}$ operations, because each of the P bracket operations requires $\mathcal{O}(N^{\frac{P+1}{P}})$.

D4 Inference with Missing Observations

The predictive mean of a Gaussian process at L test points, given N training points, is given by

$$\boldsymbol{\mu}_L = \mathbf{K}_{LN}\left(\mathbf{K}_N + \sigma^2 I_N\right)^{-1}\mathbf{y}\,, \tag{D.8}$$

where $\mathbf{K}_{LN}$ is an $L \times N$ matrix of cross covariances between the test and training points. We wish to show that when we have M observations which are not on a grid that the desired predictive mean

$$\boldsymbol{\mu}_L = \mathbf{K}_{LM}\left(\mathbf{K}_M + \sigma^2 I_M\right)^{-1}\mathbf{y}_M = \mathbf{K}_{LN}\left(\mathbf{K}_N + \mathbf{D}_N\right)^{-1}\mathbf{y}\,, \tag{D.9}$$

where $\mathbf{y} = [\boldsymbol{y}_M, \boldsymbol{y}_W]^{\top}$ includes imaginary observations $\boldsymbol{y}_W$, and D_N is as defined in section 4. as

$$D_N = \begin{bmatrix} D_M & 0 \\ 0 & \epsilon^{-1} I_W \end{bmatrix}, \tag{D.10}$$

where we set $D_M = \sigma^2 I_M$.

Starting with the right hand side of Eq. (D.9),

$$\boldsymbol{\mu}_L = \begin{bmatrix} \mathbf{K}_{LM} \\ \mathbf{K}_{LW} \end{bmatrix} \begin{bmatrix} \mathbf{K}_M + \mathbf{D}_M & \mathbf{K}_{MW} \\ \mathbf{K}_{MW}^{\top} & \mathbf{K}_W + \epsilon^{-1}\mathbf{I}_W \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{y}_M \\ \mathbf{y}_W \end{bmatrix}. \tag{D.11}$$

Using the block matrix inversion theorem, we get

$$\begin{bmatrix} A & B \\ C & E \end{bmatrix}^{-1} = \begin{bmatrix} (A - BE^{-1}C)^{-1} & -A^{-1}B(I - E^{-1}CA^{-1}B)^{-1}E^{-1} \\ -E^{-1}C(A - BE^{-1}C)^{-1} & (I - E^{-1}CA^{-1}B)^{-1}E^{-1} \end{bmatrix}, \tag{D.12}$$

where $A = \mathbf{K}_M + \mathbf{D}_M$, $B = \mathbf{K}_{MW}$, $C = \mathbf{K}_{MW}^\top$, and $E = \mathbf{K}_W + \epsilon^{-1}\mathbf{I}_W$. If we take the limit of $E^{-1} = \epsilon(\epsilon\mathbf{K}_W + \mathbf{I}_W)^{-1} \xrightarrow{\epsilon\to 0} \mathbf{0}$, and solve for the other components, Eq. (D.11) becomes

$$\boldsymbol{\mu}_L = \begin{bmatrix} \mathbf{K}_{LM} \\ \mathbf{K}_{LW} \end{bmatrix} \begin{bmatrix} (\mathbf{K}_M + \mathbf{D}_M)^{-1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} \end{bmatrix} \begin{bmatrix} \mathbf{y}_M \\ \mathbf{y}_W \end{bmatrix} = \mathbf{K}_{LM}(\mathbf{K}_M + \mathbf{D}_M)^{-1}\mathbf{y}_M \tag{D.13}$$

which is the exact GP result. In other words, performing inference given observations $\boldsymbol{y}$ will give the same result as directly using observations $\boldsymbol{y}_M$. The proof that the predictive covariances remain unchanged proceeds similarly.

References

Abrahamsen, P. (1997). A review of Gaussian random fields and correlation functions. *Norweigan Computing Center Technical report.* 117, 123

Abramowitz, M. and Stegun, I. (1964). *Handbook of Mathematical Functions with Formulas, Graphs, and Mathematical Tables.* Dover publications. 50

Adams, R. P., Dahl, G. E., and Murray, I. (2010). Incorporating side information into probabilistic matrix factorization using Gaussian processes. In *Uncertainty in Artificial Intelligence*, pages 1–9. 79

Adams, R. P. and Stegle, O. (2008). Gaussian process product models for nonparametric nonstationarity. In *International Conference on Machine Learning.* 65, 73, 74, 79, 88

Alvarez, M. and Lawrence, N. (2011). Computationally efficient convolved multiple output Gaussian processes. *JMLR*, 12:1425–1466. 64, 73, 79, 90, 91, 92

Archambeau, C. and Bach, F. (2011). Multiple Gaussian process models. *arXiv preprint arXiv:1110.5238.* 112

Asai, M., McAleer, M., and Yu, J. (2006). Multivariate stochastic volatility: a review. *Econometric Reviews*, 25(2):145–175. 175, 183

Atkinson, K. E. (2008). *An introduction to numerical analysis.* John Wiley & Sons. 61, 153

Balbach, J., Forge, V., van Nuland, N., and Winder, S. (1995). Following protein folding in real time using NMR spectroscopy. *Nature Structural and Molecular Biology*, 2(10):865–870. 103

Bartlett, M. (1933). On the theory of statistical regression. *Proceedings of the Royal Society of Edinburgh*, 53:260–283. 188, 189

Bishop, C. (1995). *Neural networks for pattern recognition.* Oxford university press. 56

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning.* Springer. 12, 15, 21, 31, 39, 40, 102

Bochner, S. (1959). *Lectures on Fourier Integrals.(AM-42)*, volume 42. Princeton University Press. 41, 114

Bollerslev, T. (1986). Generalized autoregressive conditional heteroskedasticity. *Journal of Econometrics*, 31(3):307–327. 175, 178

Bollerslev, T., Engle, R. F., and Wooldridge, J. M. (1988). A capital asset pricing model with time-varying covariances. *The Journal of Political Economy*, 96(1):116–131. 84, 181

Bonilla, E., Chai, K., and Williams, C. (2008). Multi-task Gaussian process prediction. In *Advances in Neural Information Processing Systems.* 64, 73

Bowen, S. and Hilty, C. (2010). Rapid simple injection for hyperpolarized NMR spectroscopy. *Physical chemistry chemical physics*, 12(22):5766–5770. 103

Boyle, P. and Frean, M. (2004). Dependent Gaussian processes. In *NIPS.* 64, 73, 89

Bretthorst, G. (1990). Bayesian analysis. i. parameter estimation using quadrature nmr models. *Journal of Magnetic Resonance (1969)*, 88(3):533–551. 104

Bretthorst, G. L. (1988). *Bayesian spectrum analysis and parameter estimation.* Springer. 116

Brochu, E., Cora, M., and de Freitas, N. (2010). A tutorial on Bayesian optimization of expensive cost functions, with applications to active user modeling and hierarchical reinforcement learning. *arXiv preprint 1012.2599.* 66, 69

Brooks, C., Burke, S., and Persand, G. (2001). Benchmarks and the accuracy of GARCH model estimation. *International Journal of Forecasting*, 17:45–56. 94

Brownlees, C. T., Engle, R. F., and Kelly, B. T. (2009). A practical guide to volatility forecasting through calm and storm. Available at SSRN: http://ssrn.com/abstract=1502915. 45, 94, 175, 180

Bru, M. (1991). Wishart processes. *Journal of Theoretical Probability*, 4(4):725–751. 87, 94

Cardoso, M., Salcedo, R., and Feyo de Azevedo, S. (1996). The simplex-simulated annealing approach to continuous non-linear optimization. *Computers & chemical engineering*, 20(9):1065–1080. 107

Cemgil, A. T. and Godsill, S. J. (2005). Probabilistic phase vocoder and its application to interpolation of missing values in audio signals. In *13th European Signal Processing Conference*. 116

Chatfield, C. (1989). *Time Series Analysis: An Introduction.* London: Chapman and Hall. 41, 114

Chib, S., Nardari, F., and Shephard, N. (2006). Analysis of high dimensional multivariate stochastic volatility models. *Journal of Econometrics*, 134(2):341–371. 90

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297. 21, 32

Cressie, N. (1993). *Statistics for Spatial Data (Wiley Series in Probability and Statistics).* Wiley-Interscience. 5, 20, 123

Criminisi, A., Pérez, P., and Toyama, K. (2004). Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9):1200–1212. 164

Cross, G. R. and Jain, A. K. (1983). Markov random field texture models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):25–39. 164

Csató, L. and Opper, M. (2001). Sparse representation for Gaussian process models. In *Advances in neural information processing systems*, volume 13, page 444. The MIT Press. 57, 85

Cunningham, J. P., Shenoy, K. V., and Sahani, M. (2008). Fast Gaussian process methods for point process intensity estimation. In *International Conference on Machine Learning*, pages 192–199. ACM. 61

Derin, H. and Elliott, H. (1987). Modeling and segmentation of noisy and textured images using gibbs random fields. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, (1):39–55. 164

Duane, A., Kennedy, A., Pendleton, B., and Roweth, D. (1987). Hybrid Monte Carlo. *Physics letters B*, 195(2):216–222. 37

Durrande, N., Ginsbourger, D., and Roustant, O. (2011). Additive kernels for Gaussian process modeling. *arXiv preprint arXiv:1103.4023*. 112

Duvenaud, D., Lloyd, J., Grosse, R., Tenenbaum, J., and Ghahramani, Z. (2013). Structure discovery in nonparametric regression through compositional kernel search. In *International Conference on Machine Learning*. 112, 155

Efros, A. A. and Leung, T. K. (1999). Texture synthesis by non-parametric sampling. In *Computer Vision, 1999. The Proceedings of the Seventh IEEE International Conference on*, volume 2, pages 1033–1038. IEEE. 164

Engle, R. (2002). New frontiers for ARCH models. *Journal of Applied Econometrics*, 17(5):425–446. 79, 86

Engle, R. and Kroner, K. (1995). Multivariate simultaneous generalized ARCH. *Econometric theory*, 11(01):122–150. 94

Engle, R., Nelson, D., and Bollerslev, T. (1994). ARCH models. *Handbook of Econometrics*, 4:2959–3038. 181

Engle, R., Victor, K., and Rothschild, M. (1990). Asset pricing with a factor-ARCH covariance structure: Empirical estimates for treasury bills. *Journal of Econometrics*, 45(1-2):213–237. 181

Engle, R. F. (1982). Autoregressive conditional heteroscedasticity with estimates of the variance of United Kingdom inflation. *Econometrica*, 50(4):987–1007. 174, 175

Ernst, R. R. (1992). Nuclear magnetic resonance Fourier transform spectroscopy. *Bioscience reports*, 12(3):143–187. 102

Fox, E. and Dunson, D. (2011). Bayesian nonparametric covariance regression. *Arxiv preprint arXiv:1101.2017*. 87, 97

Friedman, N. and Nachman, I. (2000). Gaussian process networks. In *Uncertainty in Artificial Intelligence*, pages 211–219. 75

Fürtig, B., Buck, J., Manoharan, V., Bermel, W., Jaschke, A., Wenter, P., Pitsch, S., and Schwalbe, H. (2007). Time-resolved NMR studies of RNA folding. *Biopolymers*, 86(5):360–383. 103

Gelfand, A., Schmidt, A., Banerjee, S., and Sirmans, C. (2004). Nonstationary multivariate process modeling through spatially varying coregionalization. *Test*, 13(2):263–312. 74, 87, 90, 97

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(2):721–741. 139, 164

Ghahramani, Z. (2013). Bayesian nonparametrics and the probabilistic approach to modelling. *Philosophical Transactions of the Royal Society A*. 19, 53, 150

Ghahramani, Z. and Beal, M. (2001). Propagation algorithms for variational Bayesian learning. *Advances in Neural Information Processing Systems*, pages 507–513. 81

Ghosal, S. and Roy, A. (2006). Posterior consistency of Gaussian process prior for nonparametric binary regression. *Annals of Statistics*, 34:2413–2429. 26, 44, 68

Gibbs, M. (1997). *Bayesian Gaussian Processes for Regression and Classification.* PhD thesis, Dept. of Physics, University of Cambridge. 48

Goldberg, P. W., Williams, C. K., and Bishop, C. M. (1998). Regression with input-dependent noise: A Gaussian process treatment. In *Advances in Neural Information Processing Systems*. 65, 73

Gönen, M. and Alpaydın, E. (2011). Multiple kernel learning algorithms. *Journal of Machine Learning Research*, 12:2211–2268. 4, 32, 112, 118, 148

Goovaerts, P. (1997). *Geostatistics for natural resources evaluation*. Oxford University Press, USA. 73, 86, 92

Gouriéroux, C. (1997). *ARCH models and financial applications*. Springer Verlag. 181

Gouriéroux, C., Jasiak, J., and Sufana, R. (2009). The Wishart autoregressive process of multivariate stochastic volatility. *Journal of Econometrics*, 150(2):167–181. 79, 86, 87, 94, 186

Granados, M., Kim, K. I., Tompkin, J., Kautz, J., and Theobalt, C. (2012). Background inpainting for videos with dynamic objects and a free-moving camera. In *European Conference on Computer Vision*. 166

Granger, C. and Andersen, A. (1978). *An Introduction to Bilinear Time-Series Models*. Vandenhoeck and Ruprecht. 174

Gray, R. M. (2006). *Toeplitz and circulant matrices: A review*. Now Publishers Inc. 59

Grimaldi, J. and Sykes, B. (1975). Stopped flow Fourier transform nuclear magnetic resonance spectroscopy. Application to the alpha-chymotrypsin-catalyzed hydrolysis of tert-butyl-l-phenylalanine. *Journal of the American Chemical Society*, 97(2):273–276. 103

Grimmett, G. and Stirzaker, D. (2001). *Probability and Random Processes*. Oxford University Press. 22

Guillemot, C. and Le Meur, O. (2014). Image inpainting: Overview and recent advances. *Signal Processing Magazine, IEEE*, 31(1):127–144. 8, 166, 167

Hansen, P. R. and Lunde, A. (2005). A forecast comparison of volatility models: Does anything beat a GARCH(1,1). *Journal of Applied Econometrics*, 20(7):873–889. 94, 175, 180

Harvey, A., Ruiz, E., and Shephard, N. (1994). Multivariate stochastic variance models. *The Review of Economic Studies*, 61(2):247–264. 84, 182

Hassner, M. and Sklansky, J. (1980). The use of markov random fields as models of texture. *Computer Graphics and Image Processing*, 12(4):357–370. 164

Hays, J. and Efros, A. (2008). Scene completion using millions of photographs. *Communications of the ACM*, 51(10):87–94. 165

Hensman, J., Fusi, N., and Lawrence, N. D. (2013). Gaussian processes for big data. In *Uncertainty in Artificial Intelligence (UAI)*. AUAI Press. 4, 57, 148

Higdon, D. (2002). Space and space-time modeling using process convolutions. In *Quantitative methods for current environmental issues*, pages 37–56. Springer. 89

Hjort, N. L., Holmes, C., Müller, P., and Walker, S. G. (2010). *Bayesian nonparametrics*. Number 28. Cambridge University Press. 19

Hore, P., Winder, S., Roberts, C., and Dobson, C. (1997). Stopped-flow photo-CIDNP observation of protein folding. *Journal of American Chemical Society*, 119(21):5049–5050. 103

Hörmander, L. (1990). *The Analysis of Linear Partial Differential Operators I, Distribution Theory and Fourier Analysis*. Springer-Verlag. 115

Hotelling, H. (1933). Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441. 181

Hutton, W. C., Bretthorst, G. L., Garbow, J. R., and Ackerman, J. J. (2009). High dynamic-range magnetic resonance spectroscopy (mrs) time-domain signal analysis. *Magnetic Resonance in Medicine*, 62(4):1026–1035. 104

Hyndman, R. (2005). Time series data library. `http://www-personal.buseco.monash.edu.au/~hyndman/TSDL/`. 129

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87. 102

Joarder, A. and Ahmed, S. (1996). Estimation of the characteristic roots of the scale matrix. *Metrika*, 44(1):259–267. 190

Joarder, A. and Ali, M. (1997). Estimation of the scale matrix of a multivariate t-model under entropy loss. *Metrika*, 46(1):21–32. 190

Jolliffe, I. (2002). *Principal component analysis*. Springer verlag. 181

Jordan, M., Ghahramani, Z., Jaakkola, T., and Saul, L. (1999). An introduction to variational methods for graphical models. *Machine learning*, 37(2):183–233. 81

Journel, A. and Huijbregts, C. (1978). *Mining geostatistics*. Academic Press (London and New York). 74, 79, 86

Julesz, B. (1962). Visual pattern discrimination. *Information Theory, IRE Transactions on*, 8(2):84–92. 165

Kass, R. E. and Greenhouse, J. B. (1989). Investigating therapies of potentially great benefit. comment: A bayesian perspective. *Statistical Science*, pages 310–317. 16

Kass, R. E. and Raftery, A. E. (1995). Bayes factors. *Journal of the American statistical association*, 90(430):773–795. 16

Keeling, C. D. and Whorf, T. P. (2004). Atmospheric CO2 records from sites in the SIO air sampling network. *Trends: A Compendium of Data on Global Change. Carbon Dioxide Information Analysis Center*. 118

Kelejian, H. and Prucha, I. (1985). Independent or uncorrelated disturbances in linear regression: An illustration of the difference. *Economics Letters*, 19(1):35–38. 191

Kersting, K., Plagemann, C., Pfaff, P., and Burgard, W. (2007). Most likely heteroscedastic Gaussian process regression. In *ICML*. 65, 73

Kirkpatrick, S., Gelatt, D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. 107

Kostantinos, N. (2000). Gaussian mixtures and their applications to signal processing. *Advanced Signal Processing Handbook: Theory and Implementation for Radar, Sonar, and Medical Imaging Real Time Systems*. 115

Kotz, S. and Nadarajah, S. (2004). *Multivariate t distributions and their applications*. Cambridge University Press. 190

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*. 2

Kühne, R., Schaffhauser, T., Wokaun, A., and Ernst, R. (1969). Study of transient chemical reactions by NMR fast stopped-flow Fourier transform experiments. *Journal of Magnetic Resonance*, 35(1):39–67. 103

Lawrence, N., Seeger, M., Herbrich, R., et al. (2003). Fast sparse gaussian process methods: The informative vector machine. *Advances in neural information processing systems*, pages 625–632. 87

Lázaro-Gredilla, M., Quiñonero-Candela, J., Rasmussen, C. E., and Figueiras-Vidal, A. (2010). Sparse spectrum Gaussian process regression. *The Journal of Machine Learning Research*, 11:1865–1881. 4, 53, 57, 116, 129, 148, 154

Lázaro-Gredilla, M. and Titsias, M. (2011). Variational heteroscedastic Gaussian process regression. In *International Conference on Machine Learning*. 65, 73

Le, Q., Sarlos, T., and Smola, A. (2013). Fastfood-computing Hilbert space expansions in loglinear time. In *International Conference on Machine Learning*, pages 244–252. 4, 50, 53, 57, 148, 155

MacKay, D. J. (1992a). Bayesian interpolation. *Neural Computation*, 4(3):415–447. 33

MacKay, D. J. (1992b). *Bayesian Methods for Adaptive Models.* PhD thesis, California Institute of Technology. 12, 13

MacKay, D. J. (1994). Bayesian nonlinear modeling for the prediction competition. *Ashrae Transactions*, 100(2):1053–1062. 117, 157

MacKay, D. J. (1998). Introduction to Gaussian processes. In Bishop, C. M., editor, *Neural Networks and Machine Learning*, chapter 11, pages 133–165. Springer-Verlag. 4, 5, 20, 40, 49, 56, 73, 148

McCulloch, W. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of mathematical biology*, 5(4):115–133. 1, 111

McCullough, B. and Renfro, C. (1998). Benchmarks and software standards: A case study of GARCH procedures. *Journal of Economic and Social Measurement*, 25:59–71. 94

Minka, T. P., Winn, J. M., Guiver, J. P., and Knowles, D. A. (2010). Infer.NET 2.4. Microsoft Research Cambridge. http://research.microsoft.com/infernet. 81

Moran, S. (2009). Video inpainting. Technical report, University of Edinburgh. 167

Murray, I. and Adams, R. (2010). Slice sampling covariance hyperparameters in latent Gaussian models. In *Advances in Neural Information Processing Systems.* 37, 38

Murray, I., Adams, R. P., and MacKay, D. J. (2010). Elliptical slice sampling. *JMLR: W&CP*, 9:541–548. 31, 38, 78, 80, 139, 141

Murray, I. and Ghahramani, Z. (2005). A note on the evidence and Bayesian Occam's razor. Technical report. 17

Naish-Guzman, A. and Holden, S. (2007). The generalized FITC approximation. In *Advances in Neural Information Processing Systems*, pages 1057–1064. 4, 148, 155

Neal, R. (1996). *Bayesian Learning for Neural Networks.* Springer Verlag. 3, 36, 45, 46, 74, 111, 112, 117

Neal, R. (2003). Slice sampling. *The Annals of Statistics*, 31(3):705–741. 37, 135, 139

Neal, R. (2010). *MCMC using Hamiltonian dynamics.* Handbook of Markov Chain Monte Carlo (ed. S. Brooks, A. Gelman, G. Jones, XL Meng). Chapman & Hall/CRC Press. 37

Nelder, J. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313. 107

Nguyen, T. V. and Bonilla, E. V. (2013). Efficient variational inference for Gaussian process regression networks. In *International Conference on Artificial Intelligence and Statistics*, pages 472–480. 72

O'Hagan, A. (1978). Curve fitting and optimal design for prediction. *Journal of the Royal Statistical Society*, B(40):1–42. 5, 20, 112

Orbanz, P. and Teh, Y. W. (2010). Bayesian nonparametric models. In *Encyclopedia of Machine Learning*, pages 81–89. Springer. 19

Osborne, M. (2010). *Bayesian Gaussian processes for sequential prediction, optimisation and quadrature.* PhD thesis, University of Oxford. 109

Patwardhan, K. A., Sapiro, G., and Bertalmio, M. (2005). Video inpainting of occluding and occluded objects. In *International Conference on Image Processing.* 166

Pearson, K. (1901). On lines and planes of closest fit to systems of points in space. *Philosophical Magazine Series 6*, 2(11):559–572. 181

Poon, S.-H. and Granger, C. W. (2005). Practical issues in forecasting volatility. *Financial Analysts Journal*, 61(1):45–56. 94, 175

Portilla, J. and Simoncelli, E. P. (2000). A parametric texture model based on joint statistics of complex wavelet coefficients. *International Journal of Computer Vision*, 40(1):49–70. 164, 165

Qi, Y., Minka, T. P., and Picara, R. W. (2002). Bayesian spectrum estimation of unevenly sampled nonstationary data. In *Acoustics, Speech, and Signal Processing (ICASSP), 2002 IEEE International Conference on*, volume 2, pages II–1473. IEEE. 104, 116

Quiñonero-Candela, J. and Rasmussen, C. E. (2005). A unifying view of sparse approximate Gaussian process regression. *The Journal of Machine Learning Research*, 6:1939–1959. 4, 57, 85, 148

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. In *In Neural Information Processing Systems*. 4, 53, 57, 148, 155

Rasmussen, C. E. (1996). *Evaluation of Gaussian Processes and Other Methods for Non-linear Regression*. PhD thesis, University of Toronto. 3, 20, 37, 66, 112

Rasmussen, C. E. and Ghahramani, Z. (2001). Occam's razor. In *Advances in Neural Information Process Systems*. 4, 12, 15, 33

Rasmussen, C. E. and Williams, C. (2006). *Gaussian processes for Machine Learning*. The MIT Press. 3, 4, 5, 20, 27, 31, 35, 43, 46, 47, 48, 49, 56, 57, 66, 69, 85, 91, 101, 112, 113, 114, 116, 117, 119, 121, 123, 124, 155, 159

Rosenblatt, F. (1962). *Principles of Neurodynamics*. Spartan Book. 1, 111

Rubtsov, D. and Griffin, J. (2007). Time-domain bayesian detection and estimation of noisy damped sinusoidal signals applied to nmr spectroscopy. *Journal of Magnetic Resonance*, 188(2):367–379. 104

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning representations by back-propagating errors. *Nature*, 323(6088):533–536. 1, 3, 111

Saatchi, Y. (2011). *Scalable Inference for Structured Gaussian Process Models*. PhD thesis, University of Cambridge. 63, 85, 149, 152, 195

Saatchi, Y., Turner, R. D., and Rasmussen, C. E. (2010). Gaussian process change point models. In *International Conference on Machine Learning*, pages 927–934. 109

Schödl, A., Szeliski, R., Salesin, D. H., and Essa, I. (2000). Video textures. In *Conference on Computer Graphics and Interactive Techniques*. 166

Seeger, M., Williams, C., and Lawrence, N. (2003). Fast forward selection to speed up sparse Gaussian process regression. In *Workshop on AI and Statistics*, volume 9, page 2003. 4, 57, 85, 148

Shah, A., Wilson, A. G., and Ghahramani, Z. (2014). Student-t processes as alternatives to Gaussian processes. *Artificial Intelligence and Statistics*. 37, 65, 69

Shumway, R. H. and Stoffer, D. S. (2006). *Time Series Analysis and Its Applications*. Springer Science and Business Media, LLC, second edition. 171

Silvennoinen, A. and Teräsvirta, T. (2009). Multivariate GARCH models. *Handbook of Financial Time Series*, pages 201–229. 175, 180, 181, 182

Snelson, E. and Ghahramani, Z. (2006). Sparse Gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, volume 18, page 1257. MIT Press. 4, 57, 85, 148, 155

Snelson, E., Rasmussen, C. E., and Ghahramani, Z. (2003). Warped Gaussian Processes. In *Advances in Neural Information Processing Systems*. 69

Stein, M. (1999). *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Verlag. 5, 20, 41, 49, 114

Steyvers, M., Griffiths, T., and Dennis, S. (2006). Probabilistic inference in human semantic memory. *Trends in Cognitive Sciences*, 10(7):327–334. 2, 112

Sutradhar, B. and Ali, M. (1986). Estimation of the parameters of a regression model with a multivariate t error variable. *Communications in Statistics-Theory and Methods*, 15(2):429–450. 191

Sutradhar, B. and Ali, M. (1989). A generalization of the Wishart distribution for the elliptical model and its moments for the multivariate t model. *Journal of Multivariate Analysis*, 29(1):155–162. 191

Teh, Y., Seeger, M., and Jordan, M. (2005). Semiparametric latent factor models. In *AISTATS*. 64, 73, 74, 75, 79, 86, 88, 100

Tenenbaum, J., Kemp, C., Griffiths, T., and Goodman, N. (2011). How to grow a mind: Statistics, structure, and abstraction. *Science*, 331(6022):1279–1285. 2, 112

Tomancak, P., Beaton, A., Weiszmann, R., Kwan, E., Shu, S., Lewis, S., Richards, S., Ashburner, M., Hartenstein, V., Celniker, S., et al. (2002). Systematic determination of patterns of gene expression during drosophila embryogenesis. *Genome Biol*, 3(12):0081–0088. 89

Tsay, R. S. (2002). *Analysis of Financial Time Series*. John Wiley & Sons. 171, 174, 178, 179, 180

Turner, R. and Sahani, M. (2007). Modeling natural sounds with modulation cascade processes. In *Advances in Neural information processing systems*. 104

Turner, R. E. (2010). *Statistical Models for Natural Sounds*. PhD thesis, University College London. 61, 65, 73, 74, 79, 88, 116, 136, 141

Uhlenback, G. and Ornstein, L. (1930). On the theory of Brownian motion. *Phys. Rev.*, 36:823–841. 50

van der Vaart, A. W. and van Zanten, J. H. (2009). Adaptive Bayesian estimation using a Gaussian random field with inverse gamma bandwidth. *Annals of Statistics*, 37:2655–2675. 44, 68

Ver Hoef, J. and Barry, R. (1998). Constructing and fitting models for cokriging and multivariable spatial prediction. *Journal of Statistical Planning and Inference*, 69(2):275–294. 88

Weierstrass, K. (1885). Über die analytische darstellbarkeit sogenannter willkürlicher functionen einer reellen veränderlichen. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin.* 16

West, G. (2013). Big Data Needs a Big Theory to Go with It. *Scientific American.* 2

Williams, C. and Rasmussen, C. E. (1996). Gaussian processes for regression. In *Advances in Neural Information Processing Systems.* 46

Williams, C. and Seeger, M. (2001). Using the Nyström method to speed up kernel machines. In *Advances in Neural Information Processing Systems*, pages 682–688. MIT Press. 4, 53, 57, 148, 154

Williams, C. K. (1998). Computation with infinite neural networks. *Neural Computation*, 10(5):1203–1216. 46, 47

Wilson, A. G. (2008). Position and energy reconstruction from scintillation light in a liquid xenon gamma ray detector designed for PET. University of British Columbia Honours Undergraduate Physics Thesis. 64

Wilson, A. G. (2012). A process over all stationary kernels. *http://mlg.eng.cam.ac.uk/andrew/procstat.pdf.* 110

Wilson, A. G. and Adams, R. P. (2013). Gaussian process kernels for pattern discovery and extrapolation. *International Conference on Machine Learning.* 110

Wilson, A. G. and Ghahramani, Z. (2010a). Copula processes. In *Advances in Neural Information Processing Systems.* 31, 65, 69, 73, 74, 98, 99, 182, 189, 192

Wilson, A. G. and Ghahramani, Z. (2010b). Generalised Wishart Processes. *Arxiv preprint arXiv:1101.0240.* 38, 65, 72, 73, 74, 77, 87, 88, 94, 97, 99, 184

Wilson, A. G. and Ghahramani, Z. (2011). Generalised Wishart Processes. In *Uncertainty in Artificial Intelligence.* 38, 65, 72, 73, 74, 77, 87, 97, 99, 184, 187

Wilson, A. G. and Ghahramani, Z. (2012). Modelling input dependent correlations between multiple responses. In *Proceedings of ECML PKDD*. 72

Wilson, A. G., Gilboa, E., Nehorai, A., and Cunningham, J. P. (2013). GPatt: Fast multidimensional pattern extrapolation with Gaussian processes. *arXiv preprint arXiv:1310.5288*. 148

Wilson, A. G., Knowles, D. A., and Ghahramani, Z. (2011). Gaussian process regression networks. *arXiv preprint arXiv:1110.4411*. 72

Wilson, A. G., Knowles, D. A., and Ghahramani, Z. (2012). Gaussian process regression networks. In *International Conference on Machine Learning*. 38, 72

Wilson, A. G., Wu, Y., Holland, D. J., Nowozin, S., Mantle, M., Gladden, L., and Blake, A. (2014). Bayesian inference for NMR spectroscopy with applications to chemical equantification. *In submission. arXiv pre-print arXiv 1402.3580*. 72, 74, 104, 107

Winn, J. and Bishop, C. M. (2006). Variational message passing. *Journal of Machine Learning Research*, 6(1):661. 81, 82

Yu, B. M., Cunningham, J. P., Santhanam, G., Ryu, S. I., Shenoy, K. V., and Sahani, M. (2009). Gaussian-process factor analysis for low-dimensional single-trial analysis of neural population activity. In *NIPS*. 64, 73, 87

Yuille, A. and Kersten, D. (2006). Vision as Bayesian inference: analysis by synthesis? *Trends in Cognitive Sciences*, 10(7):301–308. 2, 112

Zellner, A. (1976). Bayesian and non-Bayesian analysis of the regression model with multivariate Student-t error terms. *Journal of the American Statistical Association*, 71(354):400–405. 191

Zinzen, R., Girardot, C., Gagneur, J., Braun, M., and Furlong, E. (2009). Combinatorial binding predicts spatio-temporal cis-regulatory activity. *Nature*, 462(7269):65–70. 90

Zohar, S. (1969). Toeplitz matrix inversion: the algorithm of WF Trench. *Journal of the Association for Computing Machinery (JACM)*, 16(4):592–601. 61